Lecture Notes in Operations Research

Panos Pardalos
Ilias Kotsireas
William J. Knottenbelt
Stefanos Leonardos *Editors*

Mathematical Research for Blockchain Economy

4th International Conference MARBLE 2023, London, United Kingdom

Springer

Lecture Notes in Operations Research

Editorial Board Members

Ana Paula Barbosa-Povoa, *University of Lisbon, Lisboa, Portugal*
Adiel Teixeira de Almeida, *Federal University of Pernambuco, Recife, Brazil*
Noah Gans, *The Wharton School, University of Pennsylvania, Philadelphia, USA*
Jatinder N. D. Gupta, *University of Alabama in Huntsville, Huntsville, USA*
Gregory R. Heim, *Mays Business School, Texas A&M University, College Station, USA*
Guowei Hua, *Beijing Jiaotong University, Beijing, China*
Alf Kimms, *University of Duisburg-Essen, Duisburg, Germany*
Xiang Li, *Beijing University of Chemical Technology, Beijing, China*
Hatem Masri, *University of Bahrain, Sakhir, Bahrain*
Stefan Nickel, *Karlsruhe Institute of Technology, Karlsruhe, Germany*
Robin Qiu, *Pennsylvania State University, Malvern, USA*
Ravi Shankar, *Indian Institute of Technology, New Delhi, India*
Roman Slowiński, *Poznań University of Technology, Poznan, Poland*
Christopher S. Tang, *Anderson School, University of California Los Angeles, Los Angeles, USA*
Yuzhe Wu, *Zhejiang University, Hangzhou, China*
Joe Zhu, *Foisie Business School, Worcester Polytechnic Institute, Worcester, USA*
Constantin Zopounidis, *Technical University of Crete, Chania, Greece*

Lecture Notes in Operations Research is an interdisciplinary book series which provides a platform for the cutting-edge research and developments in both operations research and operations management field. The purview of this series is global, encompassing all nations and areas of the world.

It comprises for instance, mathematical optimization, mathematical modeling, statistical analysis, queueing theory and other stochastic-process models, Markov decision processes, econometric methods, data envelopment analysis, decision analysis, supply chain management, transportation logistics, process design, operations strategy, facilities planning, production planning and inventory control.

LNOR publishes edited conference proceedings, contributed volumes that present firsthand information on the latest research results and pioneering innovations as well as new perspectives on classical fields. The target audience of LNOR consists of students, researchers as well as industry professionals.

Panos Pardalos · Ilias Kotsireas ·
William J. Knottenbelt · Stefanos Leonardos
Editors

Mathematical Research for Blockchain Economy

4th International Conference MARBLE 2023 London, United Kingdom

Springer

Editors
Panos Pardalos
Department of Industrial and Systems Engineering
University of Florida
Gainesville, FL, USA

William J. Knottenbelt
Department of Computing
Imperial College London
London, UK

Ilias Kotsireas
CARGO Lab
Wilfrid Laurier University
Waterloo, ON, Canada

Stefanos Leonardos
Department of Informatics
King's College London
London, UK

ISSN 2731-040X ISSN 2731-0418 (electronic)
Lecture Notes in Operations Research
ISBN 978-3-031-48730-9 ISBN 978-3-031-48731-6 (eBook)
https://doi.org/10.1007/978-3-031-48731-6

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Paper in this product is recyclable.

Preface

This volume presents the proceedings of the 4th International Conference on Mathematical Research for Blockchain Economy (MARBLE 2023) that was held in London, United Kingdom from July 11 to 13, 2023.

The 4th MARBLE conference took place as an in-person event and featured an exciting programme of research papers, keynote talks and a tutorial, in line with MARBLE's goal to provide a high-profile, cutting-edge platform for mathematicians, computer scientists and economists to present the latest advances and innovations related to the quantitative and economic aspects of blockchain technology. In this context, the Technical Programme Committee has accepted 12 research papers for publication and presentation on themes including mining incentives, game theory, decentralized finance, central government digital coins and stablecoins, automated market makers, blockchain infrastructure and security. The technical programme also features keynotes by the following distinguished speakers: Dr. Garrick Hileman, Tara Annison (Twinstake), Artur Sepp (Clearstar), Mark Morton (Scilling Digital Mining), Dr. Jiahua Xu (University College London & DLT Science Foundation), Thomas Erdösi (CF Benchmarks), Dr. Alexander Freier (Catholic University of Cordoba, University College London and Energiequelle) and Juan Ignacio Ibañez (University College London and DLT Science Foundation).

We thank all authors who submitted their innovative work to MARBLE 2023. In addition, we thank all members of the Technical Programme Committee and other reviewers, everyone who submitted a paper for consideration, the General Chairs, Prof. William Knottenbelt and Prof. Panos Pardalos; the Organization Chair, Jas Gill; the Programme Chairs, My T. Thai and Stefanos Leonardos; the Publication Chair, Ilias Kotsireas; the Web and Publicity Chairs, Kai Sun and Gemma Ralton; and other members of the Centre for Cryptocurrency Research and Engineering who have contributed in many different ways to the organization effort, particularly Katerina Koutsouri. Finally, we are grateful to our primary sponsor, the Brevan Howard Centre for Financial Analysis, for their generous and ongoing support.

London, UK
July 2023

William J. Knottenbelt
Ilias Kotsireas
Stefanos Leonardos
Panos Pardalos

Contents

Deep Reinforcement Learning-Based Rebalancing Policies for Profit Maximization of Relay Nodes in Payment Channel Networks 1
Nikolaos Papadis and Leandros Tassiulas

Game-Theoretic Randomness for Proof-of-Stake 28
Zhuo Cai and Amir Goharshady

Incentive Schemes for Rollup Validators 48
Akaki Mamageishvili and Edward W. Felten

Characterizing Common Quarterly Behaviors in DeFi Lending Protocols 62
Aaron Green, Michael Giannattasio, Keran Wang, John S. Erickson, Oshani Seneviratne, and Kristin P. Bennett

Blockchain Transaction Censorship: (In)secure and (In)efficient? 78
Zhipeng Wang, Xihan Xiong, and William J. Knottenbelt

An Automated Market Maker Minimizing Loss-Versus-Rebalancing 95
Conor McMenamin, Vanesa Daza, and Bruno Mazorra

Profit Lag and Alternate Network Mining 115
Cyril Grunspan and Ricardo Pérez-Marco

Oracle Counterpoint: Relationships Between On-Chain and Off-Chain Market Data 133
Zhimeng Yang, Ariah Klages-Mundt, and Lewis Gudgeon

Exploring Decentralized Governance: A Framework Applied to Compound Finance 152
Stamatis Papangelou, Klitos Christodoulou, and George Michoulis

A Mathematical Approach on the Use of Integer Partitions for Smurfing in Cryptocurrencies 169
Bernhard Garn, Klaus Kieseberg, Ceren Çulha, Marlene Koelbing, and Dimitris E. Simos

Bigger Than We Thought: The Upbit Hack Gang 178
Qishuang Fu, Dan Lin, and Jiajing Wu

Staking Pools on Blockchains 187

Deep Reinforcement Learning-Based Rebalancing Policies for Profit Maximization of Relay Nodes in Payment Channel Networks

Nikolaos Papadis[1(✉)] and Leandros Tassiulas[2]

[1] Nokia Bell Labs, Murray Hill, NJ 07974, USA
nikos.papadis@nokia-bell-labs.com
[2] Department of Electrical Engineering & Yale Institute for Network Science, Yale University, New Haven, CT 06520, USA
leandros.tassiulas@yale.edu

Abstract. Payment channel networks (PCNs) are a layer-2 blockchain scalability solution, with its main entity, the payment channel, enabling transactions between pairs of nodes "off-chain," thus reducing the burden on the layer-1 network. Nodes with multiple channels can serve as relays for multihop payments by providing their liquidity and withholding part of the payment amount as a fee. Relay nodes might after a while end up with one or more unbalanced channels, and thus need to trigger a rebalancing operation. In this paper, we study how a relay node can maximize its profits from fees by using the rebalancing method of submarine swaps. We introduce a stochastic model to capture the dynamics of a relay node observing random transaction arrivals and performing occasional rebalancing operations, and express the system evolution as a Markov Decision Process. We formulate the problem of the maximization of the node's fortune over time over all rebalancing policies, and approximate the optimal solution by designing a Deep Reinforcement Learning (DRL)-based rebalancing policy. We build a discrete event simulator of the system and use it to demonstrate the DRL policy's superior performance under most conditions by conducting a comparative study of different policies and parameterizations. Our work is the first to introduce DRL for liquidity management in the complex world of PCNs.

Keywords: Payment channel networks · Lightning Network · Rebalancing · Submarine swaps · Deep reinforcement learning · Soft actor-critic · Optimization · Discrete event simulation · Control

Work done while N. Papadis was with the Department of Electrical Engineering and the Yale Institute for Network Science, Yale University, CT 06520, USA.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
P. Pardalos et al. (Eds.): MARBLE 2023, 2024.
https://doi.org/10.1007/978-3-031-48731-6_1

1 Introduction

Blockchain technology enables trusted interactions between untrusted parties, with financial applications like Bitcoin and beyond, but with also known scalability issues [8]. Payment channels are a layer-2 development towards avoiding the long confirmation times and high costs of the layer-1 network: they enable nodes that want to transact quickly, cheaply and privately to do so by depositing some balances to open a payment channel between themselves, and then trustlessly shifting the same total balance between the two sides without broadcasting their transactions and burdening the network. Connected channels create a Payment Channel Network (PCN), via which two nodes not sharing a channel can still pay one another via a sequence of existing channels. Intermediate nodes in the PCN function as relays: they forward the payment along its path and collect relay fees in return. As transactions flow through the PCN, some channels get depleted, causing incoming transactions to fail because of insufficient liquidity on their path. Thus, the need for channel rebalancing arises.

In this paper, we study the rebalancing mechanism of submarine swaps, which allows a blockchain node to exchange funds from on- to off-chain and vice versa. Since a swap involves an on-chain transaction, it takes some time to complete. Taking this into account, we formulate the following optimal rebalancing problem as a Markov Decision Process (MDP): For a node relaying traffic across multiple channels, determine an optimal rebalancing strategy over time (i.e. when and how much to rebalance as a function of the transaction arrival rates observed from an unknown distribution and the confirmation time of an on-chain transaction), so that the node can keep its channels liquid and its profit from relay fees can be maximized.

More specifically, our *contributions* are the following:

- We develop a stochastic model that captures the dynamics of a relay node with two payment channels under two timescales: a continuous one for random discrete transaction arrivals in both directions from distributions unknown to the node, and a discrete one for dispatching rebalancing operations.
- We express the system evolution in our model as an MDP with continuous state and action spaces and time-dependent constraints on the actions, and formulate the problem of relay node profit maximization.
- We approximate the optimal policy of the MDP using Deep Reinforcement Learning (DRL) by appropriately engineering the states, actions and rewards and tuning a version of the Soft Actor-Critic algorithm.
- We develop a discrete event simulator of the system, and use it to evaluate the performance of the learning-based as well as other heuristic rebalancing policies under various transaction arrival conditions and demonstrate the superiority of our policy in a range of regimes.

In summary, our paper is the first to formally study the submarine swap rebalancing mechanism and to introduce a DRL-based method for channel rebalancing in particular, and for PCN liquidity management in general.

2 Background

2.1 Payment Channel Networks and the Need for Rebalancing

A payment channel (Fig. 1) is created between two nodes N_1 and N_2 after they deposit some capital to a channel-opening on-chain transaction. After this transaction is confirmed, the nodes can transact completely *off-chain* (i.e. in the channel) without broadcasting their interactions to the layer-1 network, and without the risk of losing funds, thanks to a cryptographic safety mechanism. The sum of their two balances in the channel remains constant and is called the channel capacity. A transaction of amount α from N_1 to N_2 will succeed if the balance of N_1 at that moment suffices to cover it. In this case, the balance of N_1 is reduced by α and the balance of N_2 is increased by α.

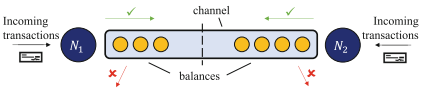


Fig. 1. A payment channel between nodes N_1 and N_2 and current balances of 3 and 4

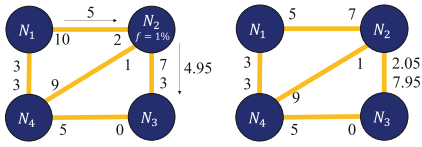


Fig. 2. Processing of a transaction in a payment channel network: before (left) and after (right)

As pairs of nodes create channels, a payment channel network (Fig. 2) is formed, over which multihop payments are possible: Consider a transaction of amount 5 from N_1 to N_3 via N_2. Note that the amount 5 includes the fees that will have be paid on the way, e.g. 1% at each intermediate node. In the N_1N_2 channel, N_1's local balance is reduced by 5 and N_2's local balance is increased by 5. In the N_2N_3 channel, N_2's local balance is reduced by $5-\text{fees} = 4.95$ and N_3's local balance is increased by 4.95. N_2's total capital in all its channels before the transaction was $2+1+7=10$, while after it is $7+1+2.05=10.05$, so N_2 made a profit of 0.05 by acting as a relay. If one of the outgoing balances did not suffice, then the transaction would fail end-to-end, thanks to a smart contract mechanism, the Hashed Time-Lock Contract (HTLC). The role of relay nodes is fundamental for the continuous operation of a PCN. The most prominent PCN currently is the Lightning Network [26] built on top of Bitcoin. More details on PCN operation can be found in [23].

Depending on the demand a payment channel is facing in its two directions, funds might accumulate on one side and deplete on the other, due to a combination of factors (see Appendix A for details). The resulting imbalance is undesirable, as it leads to transaction failures and loss of profit from relay fees, thus creating the need for rebalancing mechanisms.

2.2 The Submarine Swap Rebalancing Mechanism

In this work, we study submarine swaps, introduced in [4] and used commercially by Boltz[1] and Loop.[2] At a high level, a *submarine swap* works as follows (Fig. 3): Node N_1 owns some funds in its channel with node N_2, and some funds on-chain. At time t_0, the channel N_1N_2 is almost depleted on N_1's side (balance = 5). N_1 can start a *swap-in* by paying an amount (50) to a Liquidity Service Provider (LSP)—a wealthy node with access to both layers—via an *on-chain* transaction, and the LSP will give this amount back (reduced by a 10% swap fee, so 45) to N_1 *off-chain* via a path that goes through N_2. The final amount that is added at N_1 (and subtracted at N_2) is $45 - \varepsilon$ due to the relay fees spent on its way from the LSP. Thus, at time t_1 the channel will be almost perfectly balanced. The reverse procedure is also possible (a *reverse submarine swap* or *swap-out*) in order for a node to offload funds from its channel, by paying the server off-chain and receiving funds on-chain. More details on the submarine swap technical protocol can be found in Appendix B.

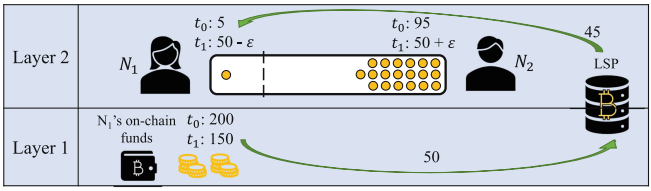


Fig. 3. A submarine swap (swap-in)

The node has to make an important tradeoff: not rebalance a lot to avoid paying swap fees but forfeit relay fees of transactions dropped due to imbalance, or vice versa. This motivates the problem of demand-aware, timely dispatching of swaps by a node aiming to maximize its total fortune.

3 Problem Formulation

3.1 System Evolution

In this section, we introduce a stochastic model of a PCN relay node N that has two channels, one with node L and one with node R, and wishes to maximize its profits from relaying payments from L to R and vice versa (Fig. 4). Let $b_{LN}(\tau), b_{NL}(\tau), b_{NR}(\tau), b_{RN}(\tau)$ be

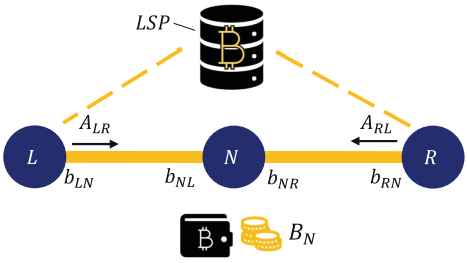


Fig. 4. System model

[1] https://boltz.exchange.
[2] https://lightning.engineering/loop.

the channel balances and $B_N(\tau)$ be the on-chain amount of N at time τ. Let C_n be the total capacity of the channel Nn, $n \in \mathcal{N} \triangleq \{L, R\}$. Events happen at two timescales: a continuous one for arriving transactions, and a discrete one for times when the node is allowed to rebalance.

The Transaction Timescale Transactions arrive as a marked point process and are characterized by their direction (L-to-R or R-to-L), time of arrival and amount. We consider node N to not be the source or destination of any transactions itself, but rather to only act as a relay. At each moment in continuous time (denoted by τ), (at most) one transaction arrives in the system. All transactions are admitted, but some fail due to insufficient balances.

Let $f(\alpha)$ be the fee that a transaction of amount α pays to a node that relays it. We assume all nodes charge the same fees. f can be any fixed function with $f(0) = 0$. In practice, for $\alpha > 0$, $f(\alpha) = f_{\text{base}} + f_{\text{prop}} \cdot \alpha$, where the base fee f_{base} and the proportional fee f_{prop} are constants. Let $A_{LR}(\tau)$, $A_{RL}(\tau)$ be the externally arriving amounts coming from node L in the L-to-R direction and from node R in the R-to-L direction at time τ respectively, each drawn from a distribution that is fixed but unknown to node N. An arriving transaction of amount $A_{LR}(\tau) = \alpha$ is feasible if and only if there is enough balance in the L-to-R direction in both channels, i.e. $b_{LN}(\tau) \geq \alpha$ and $b_{NR} \geq \alpha - f(\alpha)$, and similarly for the R-to-L direction. The successfully processed amounts by N at time τ are[3]:

$$S_{LR}(\tau) = \begin{cases} A_{LR}(\tau) & \text{, if } A_{LR}(\tau) \leq b_{LN}(\tau) \text{ and } A_{LR}(\tau) - f(A_{LR}(\tau)) \leq b_{NR}(\tau) \\ 0 & \text{, otherwise} \end{cases}$$

and symmetrically for $S_{RL}(\tau)$.

The profit of node N at time τ is $f(S_{LR}(\tau)) + f(S_{RL}(\tau))$, and the lost fees (from transactions that potentially failed to process) are $f\big(A_{LR}(\tau) - S_{LR}(\tau)\big) + f\big(A_{RL}(\tau) - S_{RL}(\tau)\big)$. The balance processes at time τ evolve as follows (the on-chain amount $B_N(\tau)$ is not affected by the processing of off-chain transactions; channel NR behaves symmetrically):

$$b_{LN}(\tau) \to b_{LN}(\tau) + (S_{RL}(\tau) - f(S_{RL}(\tau))) - S_{LR}(\tau)$$
$$b_{NL}(\tau) \to b_{NL}(\tau) + S_{LR}(\tau) - (S_{RL}(\tau) - f(S_{RL}(\tau)))$$

The Rebalancing Decision (Control) Timescale The evolution of the system can be controlled by node N using submarine swap rebalancing operations. Rebalancing may start at times $t_i = i \cdot T_{\text{check}}$, $i = 0, 1, \ldots$, and takes a (fixed) time T_{conf} to complete (on average 10 min for Bitcoin). We consider the case

[3] Since in the sequel we focus on the discrete and sparse time scale of the periodic times at which the node rebalances, we make the fair assumption (as e.g. in [2]) that off-chain transactions are processed instantaneously across their entire path and do not fail in their subsequent steps after they cross the two channels (if a transaction were to fail outside the two channels, it can be viewed as of zero value by the system).

where $T_{\text{check}} \geq T_{\text{conf}}$ (to avoid having concurrent rebalancing operations in the same channel that could be combined into one). The system state is defined only for the discrete rebalancing decision timescale as the collection of the off- and on-chain balances:

$$S(t_i) = \big(b_{LN}(t_i), b_{NL}(t_i), b_{NR}(t_i), b_{RN}(t_i), B_N(t_i)\big) \tag{1}$$

At each time t_i, node N can decide to request a swap-in or a swap-out in each channel. Call the respective amounts $r_L^{\text{in}}(t_i), r_L^{\text{out}}(t_i), r_R^{\text{in}}(t_i), r_R^{\text{out}}(t_i)$. At any time t_i, in a given channel, either a swap-in or a swap-out or nothing will be requested, but not both a swap-in and a swap-out.[4]

Let $F_{\text{swap}}^{\text{in}}(\alpha)$ and $F_{\text{swap}}^{\text{out}}(\alpha)$ be the swap fees that the LSP charges for an amount α for a swap-in and a swap-out respectively, where $F_{\text{swap}}^{\text{in}}(\cdot)$ and $F_{\text{swap}}^{\text{out}}(\cdot)$ are any nonnegative functions with $F_{\text{swap}}(0) = 0$. For ease of exposition, we let all types of fees the node will have to pay (relay fees for the off-chain part, on-chain miner fees, server fees) be both part of the above swap fees, and be the same for both swap-ins and swap-outs when a net amount r_{net} is transferred from on- to off-chain or vice versa: $F_{\text{swap}}^{\text{in}}(r_{\text{net}}) = F_{\text{swap}}^{\text{out}}(r_{\text{net}}) = F_{\text{swap}}(r_{\text{net}}) \triangleq r_{\text{net}}F + M$, where the proportional part F includes the server fee and off-chain relay fees, and M includes the miner fee and potential base fees.

Note that the semantics of the swap amounts r are such that they represent the amount that will move *in the channel* (and not necessarily the net change in the node's fortune). As a result of this convention and based on the swap operation as described in the following paragraph, the amount r^{in} of a swap-in coincides with the net amount $r_{\text{net}}^{\text{in}}$ by which the node's fortune decreases (as r^{in} does not include the swap fee), while the amount r^{out} of a swap-out includes the swap fee and the net amount by which the node's fortune decreases is $r_{\text{net}}^{\text{out}} = \phi^{-1}(r^{\text{out}})$, where $\phi(r_{\text{net}}) \triangleq r_{\text{net}} + F_{\text{swap}}(r_{\text{net}})$, and ϕ^{-1} is the generalized inverse function of $\phi(\cdot)$ (it always exists: $\phi^{-1}(y) = \min\{x \in \mathbb{N} : \phi(x) = y\}$). For our $F_{\text{swap}}(\cdot)$ it is $\phi(r_{\text{net}}) = r_{\text{net}}(1 + F) + M$ for $r_{\text{net}} > 0$, $\phi(0) = 0$, so $\phi^{-1}(y) = (y - M)/(1 + F)$ for $y > 0$ and $\phi^{-1}(0) = 0$.

A Submarine Swap Step-by-Step We now describe how a rebalancing operation on the Nn channel is affecting the system state. First, we describe a swap-in of amount r_n^{in} initiated by node N to refill N's local balance in the Nn channel:

- At time t_i, node N locks the net rebalancing amount plus fees and subtracts it from its on-chain funds: $B_N \to B_N - (r_n^{\text{in}} + F_{\text{swap}}^{\text{in}}(r_n^{\text{in}}))$
- At time $t_i + T_{\text{conf}}$, the on-chain transaction is confirmed, so the LSP sends a payment of r_n^{in} to node N off-chain.[5] The payment reaches node n:
  - If $b_{nN} \leq r_n^{\text{in}}$ (i.e. n does not have enough balance to forward it), then the off-chain payment fails. The on-chain funds are unlocked and refunded back to the on-chain amount: $B_N \to B_N + (r_n^{\text{in}} + F_{\text{swap}}^{\text{in}}(r_n^{\text{in}}))$

[4] Nodes L and R are considered passive: they perform no swap operations themselves.

[5] The LSP is a well-connected node owning large amounts of liquidity, so we reasonably assume that it can always find a route from itself to N, possibly via splitting the amount across multiple paths.

- Otherwise (if the transaction is feasible), n forwards the payment to N: $b_{nN} \to b_{nN} - r_n^{\text{in}}$ and $b_{Nn} = b_{Nn} + r_n^{\text{in}}$

A swap-out of amount r_n^{out}, initiated by node N to offload some of its local balance to the chain, works as follows:

- At time t_i, node N locks the net rebalancing amount plus fees and sends it to the LSP via the off-chain network: $b_{Nn} \to b_{Nn} - r_n^{\text{out}}$. Note that r_n^{out} includes the fees.
- At time $t_i + T_{\text{conf}}$, the on-chain transaction is confirmed, so node N receives the funds on-chain: $B_N \to B_N + \phi^{-1}(r_n^{\text{out}})$, and the funds are also unlocked in the channel and pushed towards the remote balance: $b_{nN} \to b_{nN} + r_n^{\text{out}}$

Rebalancing Constraints Based on the steps just described, swap operations will succeed if and only if their amounts satisfy the following constraints:

- Rebalancing amounts must be non-negative:

$$r_n^{\text{in}}(t_i), r_n^{\text{out}}(t_i) \geq 0 \text{ for all } i \in \mathbb{N}, n \in \mathcal{N} \tag{2}$$

- A swap-in and a swap-out cannot be requested in the same channel at the same time:

$$r_n^{\text{in}}(t_i) \cdot r_n^{\text{out}}(t_i) = 0 \text{ for all } i \in \mathbb{N}, n \in \mathcal{N} \tag{3}$$

- The swap-out amounts (which already include the swap fees) must be greater than the fees themselves:

$$r_n^{\text{out}}(t_i) - F_{\text{swap}}^{\text{out}}(r_n^{\text{out}}(t_i)) \geq 0 \text{ for all } i \in \mathbb{N}, n \in \mathcal{N} \tag{4}$$

- The respective channel balances must suffice to cover the swap-out amounts (which already include the swap fees):

$$r_n^{\text{out}}(t_i) \leq b_{Nn}(t_i) \text{ for all } i \in \mathbb{N}, n \in \mathcal{N} \tag{5}$$

- The on-chain balance must suffice to cover the total swap-in amount plus fees:

$$\sum_{n \in \mathcal{N}} \left(r_n^{\text{in}}(t_i) + F_{\text{swap}}^{\text{in}}(r_n^{\text{in}}(t_i)) \right) \leq B_N(t_i) \text{ for all } i \in \mathbb{N} \tag{6}$$

State Evolution Equations Now we are able to write the complete state evolution equations. The amounts added to each balance due to successful transactions during the interval (t_i, t_{i+1}) are:

$$d_{NL}^{(t_i, t_{i+1})} \triangleq \int_{\tau \in (t_i, t_{i+1})} \Big(S_{LR}(\tau) - (S_{RL}(\tau) - f(S_{RL}(\tau))) \Big) d\tau,$$

similarly for $d_{NR}^{(t_i,t_{i+1})}$, and $d_{nN}^{(t_i,t_{i+1})} \triangleq -d_{Nn}^{(t_i,t_{i+1})}$. Then for actions taken subject to the constraints (2)–(6), the state evolves as follows:

$$b_{nN}(t_{i+1}) = b_{nN}(t_i) + d_{nN}^{(t_i,t_{i+1})} - (r_n^{\text{in}}(t_i) - z_n(t_i)) + r_n^{\text{out}}(t_i)$$
$$b_{Nn}(t_{i+1}) = b_{Nn}(t_i) + d_{Nn}^{(t_i,t_{i+1})} + (r_n^{\text{in}}(t_i) - z_n(t_i)) - r_n^{\text{out}}(t_i)$$
$$B_N(t_{i+1}) = B_N(t_i) - \sum_{n\in\mathcal{N}} \left(r_n^{\text{in}}(t_i) - F_{\text{swap}}^{\text{in}}(r_n^{\text{in}}(t_i)) \right) + \sum_{n\in\mathcal{N}} \phi^{-1}(r_n^{\text{out}}(t_i)) + \sum_{n\in\mathcal{N}} w_n(t_i)$$

where $z_n(t_i)$ and $w_n(t_i)$ are the refunds of the swap-in amount off- and on-chain respectively in case a swap-in operation fails:

$$z_n(t_i) = r_n^{\text{in}}(t_i) \mathbb{1}\{b_{nN}(t_i) + d_{nN}^{(t_i,t_i+T_{\text{conf}})} < r_n^{\text{in}}(t_i)\} \tag{7}$$
$$w_n(t_i) = z_n(t_i) + F_{\text{swap}}^{\text{in}}(z_n(t_i)) \tag{8}$$

3.2 Writing the Problem as a Markov Decision Process

The objective function the node wishes to maximize in the real world is its total fortune both in the channels and on-chain (another equivalent objective is discussed in Appendix C). The fortune increase due to the action (the 4-tuple) $r(t_i)$ taken at step t_i is:

$$D(t_i, r(t_i)) \triangleq \left(\sum_{n\in\mathcal{N}} b_{Nn}(t_{i+1}) + B_N(t_{i+1}) \right) - \left(\sum_{n\in\mathcal{N}} b_{Nn}(t_i) + B_N(t_i) \right)$$

A control policy $\pi = \{(t_i, r^\pi(t_i))\}_{i\in\mathbb{N}}$ consists of the times t_i and the corresponding actions $r^\pi(t_i) = \left(r_L^{\text{in}}(t_i), r_L^{\text{out}}(t_i), r_R^{\text{in}}(t_i), r_R^{\text{out}}(t_i)\right)$ taken from the set of allowed actions $\mathcal{R} = [0, C_L]^2 \times [0, C_R]^2$, and belongs to the set of admissible policies

$$\Pi = \left\{ \{(t_i, r(t_i))\}_{i\in\mathbb{N}} \text{ such that } r(t_i) \in \mathcal{R} \text{ for all } i \in \mathbb{N} \right\}$$

Ultimately, the goal of node N is to find a rebalancing policy that maximizes the long-term average expected fortune increase D over all admissible policies:

$$\underset{\pi\in\Pi}{\text{maximize}} \lim_{H\to\infty} \frac{1}{t_H} \sum_{i=0}^{H} \mathbb{E}\left[D(t_i, r^\pi(t_i))\right]$$

subject to (2)–(6).

4 Heuristic and Reinforcement Learning-Based Policies

In this section, we describe the steps we took in order to apply DRL to approximately solve the formulated MDP. We first outline two heuristic policies, which we will use later to benchmark our DRL-based solution.

4.1 Heuristic Policies

Autoloop [7,18] is a policy that allows a node to schedule automatic swap-ins (resp. swap-outs) if its local balance falls below a minimum (resp. rises above a maximum) threshold expressed as a percentage of the channel's capacity.[6] The initiated swap is of amount equal to the difference of the local balance from the midpoint, i.e. the average of the two thresholds. The pseudocode can be found in Algorithm 1. We expect Autoloop to be suboptimal with respect to profit maximization in certain cases, as it does not take the expected demand into account and thus possibly performs rebalancing at times when it is not necessary.

Algorithm 1: Autoloop rebalancing policy

```
Input: state as in Eq. (1)
Parameters: T_check, low, high
every T_check do
    foreach neighbor n ∈ 𝒩 do
        midpoint = C_n · (low + high)/2
        if b_Nn < low · C_n then
            Swap-in amount = midpoint − b_Nn
        else if b_Nn > high · C_n then
            Swap-out amount = b_Nn − midpoint
        else
            Perform no action
```

This motivates us to define another heuristic policy that incorporates the empirical demand information. We call this policy Loopmax, as its goal is to rebalance with the maximum possible amount and as infrequently as possible (without sacrificing transactions), based on the demand. Loopmax keeps track of the total arriving amounts, and estimates the net change of each balance per unit time using the difference of the total amounts that arrived in each direction:

$$\hat{A}^{\text{net}}_{LN}(\tau) = -\hat{A}^{\text{net}}_{NL}(\tau) \triangleq \frac{1}{\tau}\int_{t\in[0,\tau]} \Big(A_{RL}(t) - f(A_{RL}(t)) - A_{LR}(t)\Big)dt \tag{9}$$

$$\hat{A}^{\text{net}}_{RN}(\tau) = -\hat{A}^{\text{net}}_{NR}(\tau) \triangleq \frac{1}{\tau}\int_{t\in[0,\tau]} \Big(A_{LR}(t) - f(A_{LR}(t)) - A_{RL}(t)\Big)dt \tag{10}$$

For each channel, we first calculate its estimated time to depletion ($ETTD$) or saturation ($ETTS$), depending on the direction of the net demand and the current balances, and using this time we dispatch a swap of the appropriate type not earlier than $T_{\text{check}} + T_{\text{conf}}$ before depletion/saturation, and of the maximum

[6] The original Autoloop algorithm defines the thresholds in terms of the node's inbound liquidity in a channel. We adopt an equivalent balance-centric view instead.

possible amount. The rationale is that if e.g. $ETTD \geq T_{\text{check}} + T_{\text{conf}}$, the policy can leverage this fact to postpone starting a swap until the next check time, since until then no transactions will have been dropped. If $ETTD < T_{\text{check}} + T_{\text{conf}}$ though, the policy should act now, as otherwise it will end up dropping transactions during the following interval of $T_{\text{check}} + T_{\text{conf}}$. The maximum possible swap-out is constrained by the local balance at that time, while the maximum possible swap-in is constrained by the remote balance at that time[7] and the on-chain amount: an on-chain amount of B_N can support (by including fees) a net swap-in amount of at most $\phi^{-1}(B_N)$. The pseudocode can be found in Algorithm 2. Compared to Autoloop, Loopmax has the advantage that it rebalances only when it is absolutely necessary and can thus achieve savings in swap fees. On the other hand, Loopmax's aggressiveness can lead it to extreme rebalancing decisions when traffic is quite skewed in a particular direction (e.g. it can do a swap-in of almost the full capacity, which is very likely to fail due to randomness in the transaction arrivals). A small modification we can use on top of Algorithm 2 to alleviate this is to define certain safety margins of liquidity that Loopmax should always leave intact on each side of the channel, so that incoming transactions do not find it depleted due to a large pending swap.

Algorithm 2: Loopmax rebalancing policy

Input: *state* as in Eq. (1)
Parameters: T_{check}

```
every T_check do
    Update {Â^net_Nn}_{n∈N} according to Eqs. (9)–(10)
    foreach neighbor n ∈ N do
        if Â^net_Nn < 0 then
            ETTD = b_Nn/|Â^net_Nn| /* estimated time to depletion */
            if ETTD < T_check + T_conf then
                Swap-in amount = max{φ^{-1}(B_N), b_nN} /* maximum possible
                swap in */
            else
                Perform no action
        else if Â^net_Nn > 0 then
            ETTS = b_nN/Â^net_Nn /* estimated time to saturation */
            if ETTS < T_check + T_conf then
                Swap-out amount = b_Nn /* maximum possible swap out */
            else
                Perform no action
        else
            Perform no action
```

[7] Actually, it is constrained by the remote balance at the time of the swap-in's completion. We will improve this later using estimates of future balances.

4.2 Deep Reinforcement Learning Algorithm Design

Having formulated the problem as an MDP, we now need to find an (approximately) optimal policy. The problem is challenging for a number of reasons:

- The problem dynamics are not linear.
- The state and action spaces are continuous and thus tabular approaches are not applicable.
- There are time-dependent constraints on the actions.
- Choosing to not rebalance at a specific time requires special treatment, as otherwise the zero action will be sampled from a continuous action space with zero probability.

To tackle these challenges, we resort to approximate methods, and specifically Reinforcement Learning (RL). In the standard RL framework, an agent makes decisions based on a policy represented as a probability distribution over states and actions: $p : p(s, a) \to [0, 1]$, with $p(s, a)$ being the probability that action a will be taken when the environment is in state s. Since our problem has continuous state and action spaces and the policy cannot be stored in tabular form, we need to use function approximation techniques. Neural networks serve well the role of function approximators in many applications [5,20]. Some algorithms appropriate for this type of problems are Deep Deterministic Policy Gradient (DDPG) [19] and Soft Actor-Critic (SAC) [15]. We decided to use the latter as DDPG is known to exhibit extreme brittleness and hyperparameter sensitivity [11]. We now describe our methodology around how we engineer our DRL algorithm based on the vanilla SAC in order to arrive at a solution that deals with all the above challenges.

For the RL agent's environment, we use as state the five balances (off- and on-chain) and the estimates of the remote balances at the time of the swap completion, each normalized appropriately: by the respective channel's capacity, or by a total target fortune in the on-chain amount's case. Thus, our state space is $[0, 1]^7$. As actions, instead of the 4-tuple of Sect. 3, we use an equivalent (due to (3)) 2-tuple (r_L, r_R), i.e. a single variable for each channel that can take both positive (swap-in) and negative (swap-out) values. Before the raw sampled action is applied, it undergoes some processing described in the sequel.

Raw actions are sampled from the entire continuous action space; thus the zero action will be selected with zero probability. In reality, though, performing zero rebalancing in a channel when a swap is not necessary is important for minimizing the costs, and an action the agent should learn to apply. To make the zero action selectable with positive probability, and at the same time prevent the agent from performing swaps too small in size, we force the respective applied action to be zero if the raw action coordinate is less than a threshold ρ_0 (e.g. 20%) of the channel capacity. Moreover, the vanilla SAC algorithm [15] operates on an action space that is a compact subset of $\mathbb{R}^k$ for all decision times. In our case, though, the allowed actions vary due to the time-dependent constraints (2)–(6). We therefore define the action space to be $[-1, 1]^2$, where each coordinate denotes the percentage not of the entire channel capacity, but of the maximum

amount available for the respective type of swap at that moment. We now focus on deriving these maximum amounts from the constraints.

All constraints are decoupled per channel, except for (6). However, we observe that given some traffic, mostly in the L-to-R direction or mostly in the R-to-L direction or equal in both directions, the local balances of node N will either deplete in one channel and accumulate in the other, or accumulate in both, but never both deplete. Thus, a swap-in in both channels in general will not be a good action. Therefore, for the RL solution's purposes we can split (6) into two constraints, one for each channel, with the right-hand side of each being the entire amount $B_N(t_i)$. In case the agent does take the not advisable decision of swap-ins in both channels and their sum exceeds the on-chain amount, one of the two will simply fail.

Another useful observation is that when a swap-in is about to complete time T_{conf} after it was requested, the remote balance in the respective channel needs to suffice (otherwise the swap-in will fail and a refund will be triggered as in Eqs. (7)–(8)):

$$r_n^{\text{in}}(t_i) \leq b_{nN}(t_i) + d_{nN}^{(t_i, t_i+T_{\text{conf}})} \text{ for all } i \in \mathbb{N}, n \in \mathcal{N} \tag{11}$$

We calculate an estimate $\hat{b}_{nN}(t_i + T_{\text{conf}})$ of the right-hand side of (11) based on the past history, with the details of the calculation given in Appendix D.1. Let $\rho_{min}^{\text{out}} \triangleq M/(1-F)$ be the minimum solution of (4). As long as $\rho_0 C_n \gg \rho_{min}^{\text{out}}$, which should hold in practice as ρ_{min}^{out} is very small, we can write all constraints (2)–(6), (11) in terms of the 2-tuple (r_L, r_R) as follows:

$$r_n \in \left[-b_{Nn}, \min\{\hat{b}_{nN}(t_i + T_{\text{conf}}), \phi^{-1}(B_N(t_i)), C_n\}\right], n \in \mathcal{N}$$

The described mapping of raw actions (sampled from the distribution on the entire action space) to the finally applied actions is summarized in Table 1.

Table 1. Mapping of raw actions sampled from the learned distribution to final swap amounts requested for channel Nn, $n \in \mathcal{N}$

Raw action $r_n \in [-1, 1]$	Corresponding absolute amount $\tilde{r}_n$	Final requested swap amount
$r_n < 0$	$\lvert r_n \rvert b_{Nn}$	Swap out $\tilde{r}_n \mathbb{1}\{\tilde{r}_n \geq \rho_0 C_n\}$
$r_n \geq 0$	$r_n \min\{\hat{b}_{nN}(t_i + T_{\text{conf}}), \phi^{-1}(B_N(t_i)), C_n\}$	Swap in $\tilde{r}_n \mathbb{1}\{\tilde{r}_n > \rho_0 C_n\}$

We craft the reward signal to guide the agent towards optimizing the objective: we add the node's fortune increase (3.2) until the next check time, subtract the fee losses from transactions dropped until the next check time, and also subtract a fixed penalty for every swap the algorithm initiates and which eventually fails. A high-level description of the most important components of the final learning process is given in Algorithm 3, and certain considerations on design choices and the potential practical applicability are provided in Appendices D.2 and D.3. We call the emerging policy "RebEL": Rebalancing Enabled by Learning.

5 Evaluation

In order to evaluate the performance of different rebalancing policies, we build a discrete event simulator of a relay node with two payment channels and rebalancing capabilities using Python SimPy.[8] The simulator treats each channel as a resource allowed to undergo at most one active swap at a time, and allows for parameterization of the initial balances, the transaction generation distributions (frequency, amount, number) in both directions, the different fees, the swap check and confirmation times, the rebalancing policy and its parameters.[9]

We simulate a relay node with two payment channels, each of a capacity of \$1000 split equally between the nodes. Transactions arrive from both sides as Poisson processes. We evaluate policies Autoloop, Loopmax and RebEL defined in Sect. 4, as well as the *None* policy that never performs any rebalancing. We

Algorithm 3: RL algorithm for RebEL policy

Input: *state* as in Eq. (1)
Parameters: T_{check}, various learning parameters, penalty

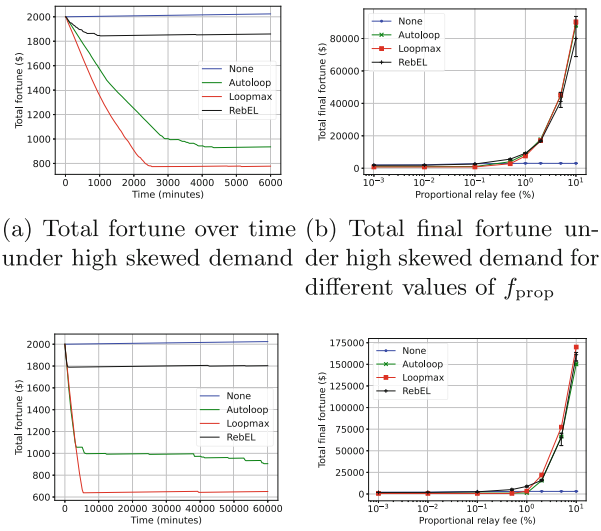
1 **every** T_{check} **do**
2 Update estimates $\hat{S}_{LR}$, $\hat{S}_{RL}$ and $\hat{b}_{LN}$, $\hat{b}_{RN}$ according to Eqs. (13)–(14)
3 Perform SAC gradient step to update policy distribution as in [15] based on replay memory
4 Fetch $state \in [0,1]^7$
5 Sample $rawAction$ from $[-1,1]^2$ according to policy distribution
6 $processedAction = process(rawAction)$ where $process(\cdot)$ is described in Table 1
7 Apply $processedAction$ and wait for its completion
8 $reward$ = fortuneAfter − fortuneBefore − lostFees − penalty · numberOfFailedSwaps
9 Fetch $nextState \in [0,1]^7$
10 Store transition $(state, rawAction, reward, nextState)$ to replay memory

[8] https://simpy.readthedocs.io.
[9] The code is publicly available at https://github.com/npapadis/payment-channel-rebalancing.

use $T_{\text{check}} = T_{\text{conf}} = 10$ minutes, miner fee $M = \$2$/on-chain transaction (tx), base fee[10] $f_{\text{base}} = 0$, swap fee $F = 0.5\%$, 0.3 and 0.7 as the low and high liquidity thresholds of Autoloop, and 2 min worth of estimated traffic as safety margins for Loopmax. We run all experiments on a regular consumer laptop.

We experimented with different hyperparameters for the original SAC algorithm[11] as well as for RebEL parameters and reward shapes, and settled with the ones shown in Appendix E. We performed experiments for the transaction amount distribution being Uniform in $[0, 50]$ and Gaussian with mean 25 and standard deviation 20, and the results were very similar. Therefore, all plots shown below are for the Gaussian amounts.

The Role of Fees Current median fee rates for transaction forwarding are in the order of $3 \cdot 10^{-5}$ (\$/\$) or 0.003%, while swap server fees are in the order of 0.5% and miner fees are in the order of 2 \$/tx.[12] In order to see if a relay node can make a profit with such fees, we perform the following back-of-the-envelope calculation: A swap-in of amount r will cost the node $rF + M$ in fees and will enable traffic of at most value r to be processed, which will yield profits rf_{prop} from relay fees. Therefore, the swap-in cannot be profitable if $rF + M \geq rf_{\text{prop}}$. Solving this inequality, we see that no positive amount r can be profitable if $f_{\text{prop}} \leq F$, while if $f_{\text{prop}} > F$ a necessary (but not sufficient) condition for profitability is $r > M/(f_{\text{prop}} - F)$. The respective inequality for a swap-out of amount r is $r - \frac{r-M}{1+F} \geq rf_{\text{prop}}$, which shows that for $f_{\text{prop}} \leq \frac{F}{1+F}$ no amount can be profitable and for $f_{\text{prop}} > \frac{F}{1+F}$ a necessary



(a) Total fortune over time under high skewed demand

(b) Total final fortune under high skewed demand for different values of f_{prop}

(c) Total fortune over time under low skewed demand

(d) Total final fortune under low skewed demand for different values of f_{prop}

Fig. 5. Experiments with different relay fee f_{prop}

[10] Currently, according to https://lnrouter.app/graph/zero-base-fee, almost 50% of the Lightning Network uses $f_{\text{base}} = 0$.

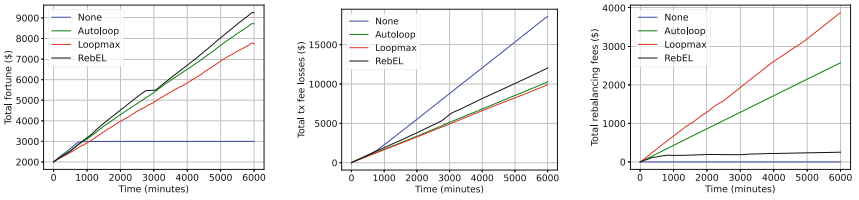
[11] We used the PyTorch implementation in https://github.com/pranz24/pytorch-soft-actor-critic.

[12] Fee value sources: https://1ml.com/statistics, https://lightning.engineering/loop, https://ycharts.com/indicators/bitcoin_average_transaction_fee.

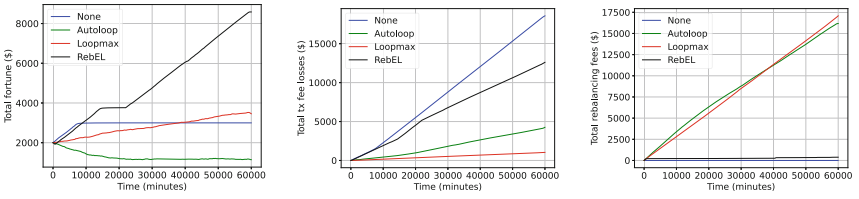
condition for profitability is that $r > \frac{M}{f_{\text{prop}}(1+F)-F}$. *With the current fees, we are in the non-profitable regime.* Although the above inequalities are short-sighted in that they focus only on a specific action time, they do confirm the observation made by both the Lightning and the academic communities [6] that in order for relay nodes to be a profitable business, relay fees have to increase.

We now perform an experiment confirming this finding with the currently used fee values. We simulate a workload of demand in the L-to-R direction: 60000 L-to-R and 15000 R-to-L transactions under a high (10 tx/minute L-to-R, 2.5 tx/minute R-to-L) and a low (1 tx/minute L-to-R, 0.25 tx/minute R-to-L) intensity. The node's total fortune over time for high and low intensity are shown in Fig. 5a and c respectively. We see that regardless of the (non-*None*) rebalancing policy, the node's fortune decreases over time, because rebalancing fees surpass any relay profits, which are small because of the small f_{prop} compared to F. In this regime, the node is better off not rebalancing at all. Still, our RebEL policy manages to learn this fact and after some point exhibits the desired behavior and stops rebalancing as well. Autoloop and Loopmax keep trying to rebalance and end up exhausting their entire on-chain balance, so the total fortune under them gets stuck after some point.

Taking a higher level view, we also conduct multiple experiments with the same demand as before but now while varying f_{prop}. The results of the total final fortune of each experiment (run for the same total time and averaged over 10 runs; error bars show the maximum and minimum values) are shown in Fig. 5b under high demand and in Fig. 5d under low demand. We see that no rebalancing



(a) Total fortune over time (high demand) (b) Transaction fee losses over time (high demand) (c) Rebalancing fees over time (high demand)



(d) Total fortune over time (low demand) (e) Transaction fee losses over time (low demand) (f) Rebalancing fees over time (low demand)

Fig. 6. Total fortune, transaction fee losses and rebalancing fees over time under demand skewed in the L-to-R direction

policy is profitable (i.e. better than *None*) as long as $f_{\text{prop}} < 0.5\% = F$, which confirms our back-of-the-envelope calculation. For higher values of f_{prop}, the node is able to make a profit. Although RebEL performs better for $f_{\text{prop}} = 1\%$ for reasons discussed in Sect. 5, Autoloop and Loopmax sometimes perform better for even higher (and thus even farther from the current) fees, because the RebEL policy used in this experiment is the one we tuned to operate best for the experiments of the next section that use $f_{\text{prop}} = 1\%$. In principle though, with different tuning, RebEL could outperform the other policies for higher values of f_{prop} as well.

The Role of the Demand We now stay in the fee regime of possible profitability by keeping $f_{\text{prop}} = 1\%$, and study the role of the demand (and indirectly of the depletion frequency) on the performance of the different policies. The results for the same high and low workload of skewed demand in the L-to-R direction as before are shown in Fig. 6. RebEL outperforms all other policies under both demand regimes (Fig. 6a, d), as it manages to strike a balance in terms of frequency and amount of rebalancing and transaction fee profits. This happens in a few 10-minute iterations under high demand (corresponding to a few hours in real time), because balance changes are more pronounced in this case and help RebEL learn faster, while it takes about 1200 iterations under low demand, translating in 8.3 days of training. Both these training times are reasonable for a relay node investing its capital to make a profit. We see that under both regimes the system without rebalancing (*None* policy) at some point reaches a state where almost all the balances are accumulated locally and no transactions can be processed anymore (hence the flattening in the *None* curve). Under high demand, Autoloop and Loopmax rebalance a lot (Fig. 6c) in order to minimize transaction fee losses (Fig. 6b), while RebEL sacrifices some transactions to achieve higher total fortune. Under low demand, RebEL rebalances only when necessary (Fig. 6f), even if this means sacrificing many more transactions (Fig. 6e), simply because rebalancing is not worth it at that low demand regime, in the sense that the potential profits during the 10-minute rebalancing check times are too low to justify the frequent rebalancing operations that the other policies apply. Loopmax eventually achieves a profit (although much lower than RebEL) because it tends to rebalance with higher amounts. On the contrary, Autoloop rebalances with small amounts, thus incurring significant costs from constant miner fees and eventually even making a loss compared to the initial node's fortune (Fig. 6d). Under high demand, there is a point around time 2700 where RebEL stalls for a bit, and the same happens under low demand between times 14000–22000. Upon more detailed inspection, this happens because all balances temporarily accumulate on the local sides of the channels. RebEL takes some steps to again bring the channels to some balance (either actively by making a swap or passively by letting transactions flow) and subsequently completely recovers.

In the special case of equal demands from both sides, RebEL does not perform as well (experiments included in Appendix F.1). However, although possible, this scenario is much less likely to occur in practice, as usually the traffic follows

some patterns, e.g. from clients to merchants. The skewed demand scenario, where RebEL is superior, is also the most natural. We also examined how the initial conditions (capacities, initial balances) affect the performance, and see that under skewed demand RebEL continues to perform well in all cases. The interested reader is referred to Appendix F.2 for the experimental evidence.

6 Related Work

Rebalancing via payments from a node to itself via a circular path of channels has been studied by several works (e.g. [1,17,25]), with some taking relay fees into account (as we did) and some not. References [10,13] describe fee strategies that incentivize the balanced use of payment channels. Reference [2] uses a game-theoretic lens to study the extent to which nodes can pay lower transaction fees by waiting patiently and reordering transactions instead of pursuing maximum efficiency. Perhaps the only work on submarine swaps is [12], which considers the problem of the appropriate fee design by liquidity providers according to usage patterns. A recent development similar to submarine swaps is PeerSwap [28]: instead of buying funds from an LSP, a node can exchange funds on-/off-chain with its channel neighbor directly. *Splicing* is another mechanism that replaces a channel with a new one with a different capacity while allowing transactions to flow in the meantime [27].

Stochastic modeling and optimization in the blockchain space has been used both in layer-1 [9,14,21,22] for performance characterization, and in layer-2 for routing [29] and scheduling [24] of payments. Deep Reinforcement Learning has been broadly applied to approximately solve challenging optimization problems from various areas and to build systems that learn to manage resources directly from experience. For example, [20] applies DRL to the resource allocation problem of packing tasks under multiple resource demands, while [3] uses DRL to solve the MDP modeling selfish mining attacks in Bitcoin-like blockchains.

7 Conclusion

In this paper, we studied the problem of relay node profit maximization using submarine swaps, and demonstrated the feasibility of applying state-of-the-art DRL techniques for solving it. Future work can explore swap-based rebalancing in a network setting, the comparison of different rebalancing methods, and the incentive problems arising. We hope that this research will inspire further interest in designing capital management strategies in the complex world of PCNs based on learning from experience as an alternative to currently applied heuristics, and will be a step towards guaranteeing the profitability of the relay nodes and, consequently, the viability and scalability of the PCNs they sustain.

Acknowledgments. This work was supported by a grant from JP Morgan Chase. The authors would like to thank Leonidas Georgiadis, Nicholas Nordlund and Konstantinos Poularakis for helpful discussions.

A Causes of Channel Depletion

Channel depletion might happen due to a number of reasons:

- Asymmetric demand inside single channels
- The random nature of arrivals causing temporary depletions at specific times (e.g. when a large transaction arrives)
- Symmetric demand between two endpoints of a multihop path which can cause imbalance due to fees withheld by intermediate nodes

An example of the third and more subtle case is given in Fig. 7, which shows the evolution over time of a subnetwork of three channels with symmetric demand of amount 20 arriving alternately from either side of the path. When each transaction is relayed by node B, a 50% fee is withheld and the remaining amount of 10 is forwarded to the next channel in the path. We see that even though the end-to-end path demand is symmetric, after a few steps the channels get unbalanced and stop being able to process any more transactions.[13]

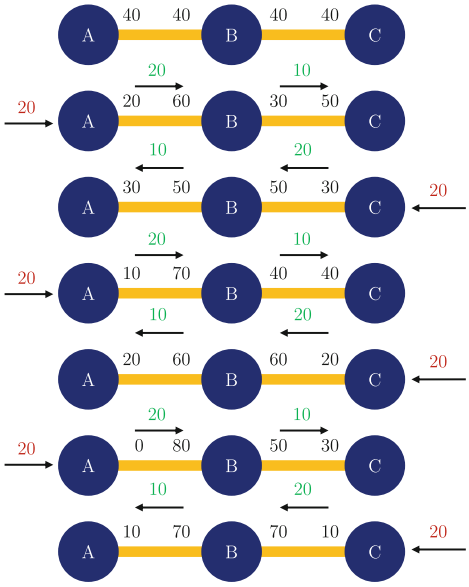


Fig. 7. An example of a PCN getting stuck even though the demand is symmetric. Demand is shown in red, forwarded amounts after a 50% fee withholding are shown in green, and channel balances are shown in black

[13] The 50% fee is not realistic and is only used for the purposes of this example. With the real much lower fees the channels will similarly get stuck after a larger number of steps.

B The Submarine Swap Protocol

A sketch of the technical protocol followed during a successful swap-in, which is the basis for the modeling of Sect. 3.1, is shown in Fig. 8 (a swap-out is similar). First, a node-client initiates the swap by generating a hash preimage, creating an invoice of the desired swap amount r tied to this hash and with a certain expiration time T_{exp}, and sends it to an LSP that is willing to make the exchange. The LSP then quotes what it wants to be paid on-chain in exchange for paying the client's invoice off-chain, say $\alpha + F_{\text{swap}}(\alpha)$, where $F_{\text{swap}}(\alpha)$ is the LSP's swap service fee. If the client accepts the exchange rate, it creates a conditional on-chain payment of amount $\alpha + F_{\text{swap}}(\alpha)$ to the LSP based on an HTLC with the same preimage as before and broadcasts the payment to the blockchain network. The payment can only be redeemed if the LSP knows the preimage, and the client will only reveal the preimage once it has received the LSP's funds on-chain. Thus, the LSP pays the off-chain invoice. This forces the client to reveal the preimage, and now the LSP can redeem the on-chain funds and the swap is complete. The entire process happens trustlessly thanks to the HTLC mechanics. More technical details can be found in [16].

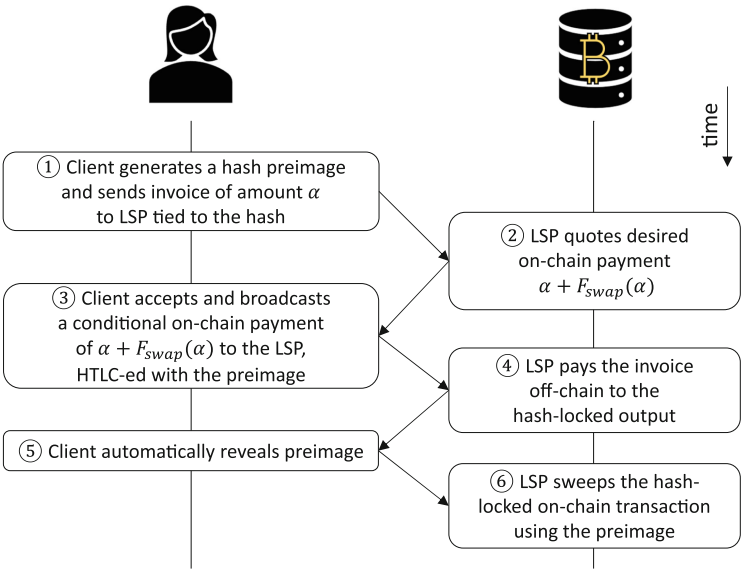


Fig. 8. A swap-in step-by-step

C An Equivalent Objective

In Sect. 3.2, we formulated the optimal rebalancing problem as a maximization of the total fortune increase. Equivalently, the node can minimize the total fee cost,

which comes from two sources: from lost fees because of dropped transactions,[14] and from fees paid for rebalancing operations:

$$L(t_i, r(t_i)) = \int_{\tau \in (t_i, t_{i+1})} \big(f(A_{LR}(\tau) - S_{LR}(\tau)) + f(A_{RL}(\tau) - S_{RL}(\tau))\big) d\tau + \sum_{n \in \mathcal{N}} \big(F^{\text{in}}_{\text{swap}}(r^{\text{in}}_n(t_i)) + F^{\text{out}}_{\text{swap}}(r^{\text{out}}_n(t_i))\big)$$

Under this objective, the ultimate goal of node N is to find a rebalancing policy that minimizes the long-term average expected fee cost L over all admissible policies:

$$\underset{\pi \in \Pi}{\text{minimize}} \lim_{H \to \infty} \frac{1}{t_H} \sum_{i=0}^{H} \mathbb{E}\left[L(t_i, r^{\pi}(t_i))\right]$$

subject to the constraints (2)–(6).

The two objectives at each timestep sum to $\int_{\tau \in (t_i, t_{i+1})} (f(A_{LR}(\tau) + f(A_{RL}(\tau)) d\tau$ (the fees that would be collected by the node if the total arriving amount had been processed), a quantity independent of the control action, and therefore maximizing the total fortune and minimizing the total fee cost are equivalent.

D Deep Reinforcement Learning Algorithm Design Details

D.1 Helping a Swap-In Succeed

When a swap-in is about to complete time T_{conf} after it was requested, the remote balance in the respective channel needs to suffice[15] (otherwise the swap-in will fail and a refund will be triggered as in Eqs. (7)–(8)):

$$r^{\text{in}}_n(t_i) \leq b_{nN}(t_i) + d_{nN}^{(t_i, t_i + T_{\text{conf}})} \text{ for all } i \in \mathbb{N}, n \in \mathcal{N} \tag{11}$$

Although (11) are not hard constraints when the decision is being made like the ones of Sect. 3.1, we would like to guide the agent to respect them. An obstacle is that the swap-in decision is made at time t_i, when the node does not yet know

[14] Note that we assume the node knows not only about the transactions that reach it, but also about the transactions that are supposed to reach it but never do because of insufficient remote balances. This is not strictly true in practice, but the node can approximate it by observing the transactions during an interval in which the remote balances are both big enough so that no incoming transaction would fail, and create an estimate based on this observation.

[15] Note that we have given the RL agent more flexibility compared to Autoloop and Loopmax: it can perform swap-ins of amount bigger than the current remote balance under the expectation that by the time of completion the balance will be adequate.

the arriving amount $d_{nN}^{(t_i,t_i+T_{\text{conf}})}$. To approximate the right-hand side of (11) in terms of quantities known at time t_i, we can use the difference of the total (and not the successful as in d_{nN}'s definition) amounts that arrived in each direction from Eqs. (9)–(10):

$$\begin{aligned} b_{nN}(t_i) + d_{nN}^{(t_i,t_i+T_{\text{conf}})} &\approx \hat{b}_{nN}(t_i + T_{\text{conf}}) \\ &\triangleq \left(\min\{b_{nN}(t_i) + \hat{A}_{nN}^{\text{net}} \cdot T_{\text{conf}}, C_n\}\right)^+ \end{aligned} \tag{12}$$

A better estimate can be obtained by using the empirical amounts that succeeded in either direction:

$$\hat{S}_{LR}(\tau) \triangleq \frac{1}{\tau}\int_{t\in[0,\tau]} S_{LR}(t)dt \qquad \text{and} \qquad \hat{S}_{RL}(\tau) \triangleq \frac{1}{\tau}\int_{t\in[0,\tau]} S_{RL}(t)dt \tag{13}$$

Then the amount $\hat{S}_{LR}$ (resp. $\hat{S}_{RL}$) will be flowing in the L-to-R (resp. R-to-L) direction for either the entire duration of T_{conf}, or until one of the balances in the respective direction is depleted:

$$\begin{aligned} \hat{b}_{LN}(t_i{+}T_{\text{conf}}) \triangleq \Bigg(\min \Bigg\{ & b_{LN}(t_i) - \hat{S}_{LR}(t_i) \min\left\{T_{\text{conf}}, \frac{b_{LN}}{\hat{S}_{LR}(t_i)}, \frac{b_{NR}}{\hat{S}_{LR}(t_i)}\right\} \\ & + (1-f_{\text{prop}})\hat{S}_{RL}(t_i) \min\left\{T_{\text{conf}}, \frac{b_{RN}}{\hat{S}_{RL}(t_i)}, \frac{b_{NL}}{\hat{S}_{RL}(t_i)}\right\}, C_L \Bigg\} \Bigg)^+ \end{aligned} \tag{14}$$

and symmetrically for the NR channel. Thus, the approximate version of (11) becomes:

$$r_n^{\text{in}}(t_i) \leq \hat{b}_{nN}(t_i + T_{\text{conf}}) \text{ for all } i \in \mathbb{N}, n \in \mathcal{N} \tag{15}$$

D.2 Design Choices

In our model in Sect. 3.1, we have considered the time for on-chain transaction confirmation and thus also rebalancing completion to be constant. In practice, completion happens when the miners solve the random puzzle and produce the Proof-of-Work for the next block that includes the rebalancing transaction. The time for this to happen fluctuates, though only slightly, so we use a fixed value for the sake of tractability.

Also, in practice, the on-chain funds used in a swap are unlocked after a time T_{exp} to prevent malicious clients from requesting many swaps from an LSP and then defaulting. However, since we are concerned with online and cooperative

clients with on-chain amounts usually quite larger than the amounts in their channels (and thus than their swaps), and also there is currently a community effort to reduce or even eliminate T_{exp}, we ignore it.

The objective in Sect. 3.2 was defined as a long-term expected average in order to match what a relay node would intuitively want to optimize, while the SAC algorithm works for long-term discounted objectives with a discount factor (usually set very close to 1), and including a maximum entropy term to enhance exploration.[16] We expect this difference to not be significant, and indeed the results show that the SAC-based policy performs well in practice.

In Sect. 5, we presented results for specific parameters and rewards for the RL algorithm. Further tuning specific to the demand regime might lead to even higher returns for the RebEL policy. Additionally, improving the estimates of future balances by having the agent perform a "mini-simulation" of the transactions arriving in the following time interval based on past statistics could help the policy produce more informed decisions. Techniques from Model Predictive Control could also be applied.

Theoretically, a class of policies that could result in even higher fortune than the class (3.2) would be one that would allow rebalancing to happen at any point in continuous time instead of periodically. Optimization in such a model however would be extremely difficult, as an action taken now would affect the state both now and in the future (when rebalancing completes). Considering that practical policies like Autoloop applied today only check for rebalancing periodically, we follow the same path for the sake of tractability.

D.3 Practical Applicability

An actual PCN node could use our simulator with samples from its past demand, and try to tune the RL parameters and the reward to get better performance than the heuristic policies we defined or the one it is currently using; then, it would apply the policy learned in the simulator environment to the real node. Alternatively, a node may not use a simulator at all and directly learn a pre-parameterized policy on the fly from the empirical transaction data. In either case, the node can do occasional retraining with updated data to account for time-variance in the distribution of the arriving demand.

[16] The exact formula for the SAC objective can be found in Appendix A of [15].

E Hyperparameters and Rewards

Table 2. SAC hyperparameters used for the different experiments of Sect. 5

SAC hyperparameter	Parameter value for skewed demand experiments	Parameter value for even demand experiments
Policy	Gaussian	
Optimizer	Adam	
Learning rate	0.0003	0.006
Discount	0.99	
Replay buffer size	10^5	
Number of hidden layers (all neural networks)	2	
Number of hidden units per layer	256	
Number of samples per minibatch	10	
Temperature	0.05	0.005
Nonlinearity	ReLU	
Target smoothing coefficient	0.005	
Target update interval	1	
Gradient steps	1	
Automatic entropy tuning	False	True
Initial random steps	10	

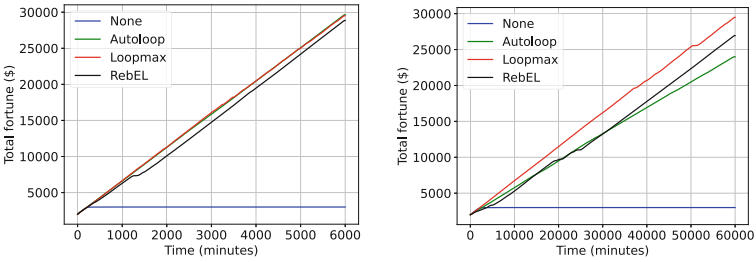
Table 3. Parameters used in RebEL's representation or processing of the states, actions, and rewards

RebEL parameter	Parameter value for skewed demand experiments	Parameter value for even demand experiments
on-chain amount normalization constant	60	
Minimum swap threshold ρ_0	0.2	
Penalty per swap failure	0	10

F Additional Experimental Results

F.1 The RebEL Policy Under Even Demand

In this section, we explore the special case of equal demands in the two directions, by applying 60000 transactions arriving on each side in high (10 tx/minute) and low (1 tx/minute) intensity. Tuning some hyperparameters and making the penalty for failed swaps non-zero as shown in Appendix E gave better results for even demand specifically, so we use this configuration for the results of Fig. 9.



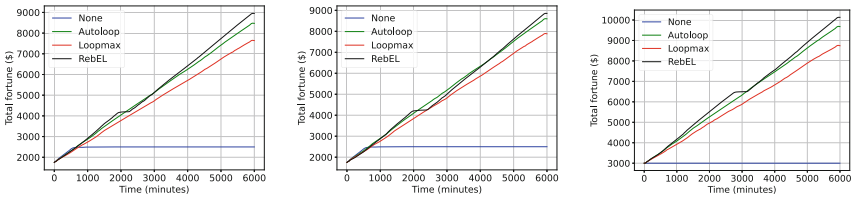
(a) High demand (b) Low demand

Fig. 9. Total fortune over time under equal demand intensity from both sides

We observe that all policies (except *None*) achieve higher total fortunes than before. This happens because the almost even traffic automatically rebalances the channel to some extent and therefore more fees can be collected in both directions and for larger amounts of time before the channels get stuck. RebEL is not as good for even traffic, because the net demand constantly oscillates around zero and this does not allow the agent to learn a good policy. It still manages though to surpass Autoloop pretty quickly under low demand, while if we run the simulation for longer times (not shown in the figure), we see that after time 78000 RebEL surpasses Loopmax as well. This translates to about 54 days of operation, which is a big time interval in practice, but is justified by the fact that the traffic is low and therefore more time is needed in order for the node to make a profit. However, even demand from both sides is a special case that is not likely to occur in practice, as usually the traffic follows some patterns, e.g. from clients to merchants. So the skewed demand scenario, where RebEL is superior, is also the most natural.

F.2 The Role of the Initial Conditions

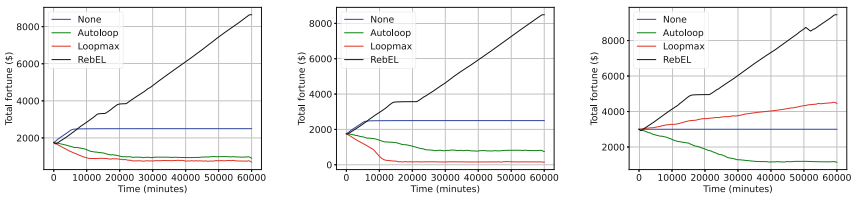
In this section, we examine how the initial conditions (capacities, initial balances) affect the performance. We evaluate all rebalancing policies for the skewed demand in the L-to-R direction scenario as before, but this time for channels of uneven capacities or initial balances. The results for high and low demand are



(a) Total fortune over time when $C_L = 1000$, $C_R = 500$ (high demand)

(b) Total fortune over time when $C_L = 500$, $C_R = 1000$ (high demand)

(c) Total fortune over time when initial balances are only local (high demand)



(d) Total fortune over time when $C_L = 1000$, $C_R = 500$ (low demand)

(e) Total fortune over time when $C_L = 500$, $C_R = 1000$ (low demand)

(f) Total fortune over time when initial balances are only local (low demand)

Fig. 10. Total fortune, transaction fee losses and rebalancing fees over time under demand skewed in the L-to-R direction for different initial conditions

shown in Fig. 10a and d respectively for $C_L = 1000$, $C_R = 500$ and the initial balances evenly distributed, in Fig. 10b and c respectively for $C_L = 500$, $C_R = 1000$ and the initial balances evenly distributed, and in Fig. 10c and f respectively for $C_L = C_R = 1000$ but $b_{NL} = b_{NR} = 1000$ (and so $b_{LN} = b_{RN} = 0$). We see that RebEL performs well in all these cases as well. Depending on the exact arriving transactions, the little plateaus of RebEL happen at different points in time for the same reason as in Fig. 6a and d, but in the end the learning algorithm recovers.

References

1. Avarikioti, Z., Pietrzak, K., Salem, I., Schmid, S., Tiwari, S., Yeo, M.: HIDE & SEEK: Privacy-preserving rebalancing on payment channel networks. In: Eyal, I., Garay, J. (eds.) Financial Cryptography and Data Security, pp. 358–373. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-18283-9_17
2. Bai, Q., Xu, Y., Wang, X.: Understanding the benefit of being patient in payment channel networks. IEEE Trans. Netw. Sci. Eng. **9**(3), 1895–1908 (2022). https://doi.org/10.1109/TNSE.2022.3154408
3. Bar-Zur, R., Abu-Hanna, A., Eyal, I., Tamar, A.: WeRLman: To tackle whale (transactions), go deep (RL). In: Proceedings of the 15th ACM International Conference on Systems and Storage, p. 148. SYSTOR '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3534056.3535005

4. Bosworth, A.: Submarine swaps on the Lightning Network (2018). https://submarineswaps.github.io
5. Boute, R.N., Gijsbrechts, J., van Jaarsveld, W., Vanvuchelen, N.: Deep reinforcement learning for inventory control: A roadmap. Eur. J. Oper. Res. **298**(2), 401–412 (2022). https://doi.org/10.1016/j.ejor.2021.07.016
6. Béres, F., Seres, I.A., Benczúr, A.A.: A cryptoeconomic traffic analysis of Bitcoin's Lightning Network. Cryptoeconomic Syst. (6) (2020). https://cryptoeconomicsystems.pubpub.org/pub/b8rb0ywn
7. Carla Kirk-Cohen: Autoloop: Lightning Liquidity You Can Set and Forget! (2020). https://lightning.engineering/posts/2020-11-24-autoloop
8. Croman, K., Decker, C., Eyal, I., Gencer, A.E., Juels, A., Kosba, A., Miller, A., Saxena, P., Shi, E., Sirer, E.G., Song, D., Wattenhofer, R.: On scaling decentralized blockchains. In: Clark, J., Meiklejohn, S., Ryan, P.Y., Wallach, D., Brenner, M., Rohloff, K. (eds.) Financial Cryptography and Data Security, pp. 106–125. Springer, Berlin Heidelberg, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_8
9. Dembo, A., Kannan, S., Tas, E.N., Tse, D., Viswanath, P., Wang, X., Zeitouni, O.: Everything is a race and Nakamoto always wins. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, p. 859–878. CCS '20, ACM (2020). https://doi.org/10.1145/3372297.3417290
10. Di Stasi, G., Avallone, S., Canonico, R., Ventre, G.: Routing payments on the Lightning Network. In: iThings/GreenCom/CPSCom/SmartData, pp. 1161–1170. IEEE (2018). https://doi.org/10.1109/Cybermatics_2018.2018.00209
11. Duan, Y., Chen, X., Houthooft, R., Schulman, J., Abbeel, P.: Benchmarking deep reinforcement learning for continuous control. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning, vol. 48, p. 1329–1338. ICML'16, JMLR.org (2016). https://doi.org/10.48550/ARXIV.1604.06778
12. Echenique, J.I.R., Burtey, N.: Pricing liquidity for Lightning wallets (2022). https://github.com/GaloyMoney/liquidity-fees-paper
13. van Engelshoven, Y., Roos, S.: The merchant: Avoiding payment channel depletion through incentives. In: IEEE International Conference on Decentralized Applications and Infrastructures, DAPPS 2021, Online Event, 23–26 Aug 2021, pp. 59–68. IEEE (2021). https://doi.org/10.1109/DAPPS52256.2021.00012
14. Gaži, P., Kiayias, A., Russell, A.: Tight consistency bounds for Bitcoin. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp. 819–838. CCS '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3372297.3423365
15. Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., Levine, S.: Soft actor-critic algorithms and applications (2018). https://doi.org/10.48550/ARXIV.1812.05905
16. Jager, J.: Loop Out in-depth (2019). http://www.blog.lightning.engineering/technical/posts/2019/04/15/loop-out-in-depth.html
17. Khalil, R., Gervais, A.: Revive: Rebalancing off-blockchain payment networks. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30–November 03, 2017, pp. 439–453. ACM (2017). https://doi.org/10.1145/3133956.3134033
18. Lightning Labs: Autoloop (2022). http://www.github.com/lightninglabs/loop/blob/master/docs/autoloop.md

19. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: Bengio, Y., LeCun, Y. (eds.) 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, 2–4 May 2016, Conference Track Proceedings (2016). https://doi.org/10.48550/ARXIV.1509.02971
20. Mao, H., Alizadeh, M., Menache, I., Kandula, S.: Resource management with deep reinforcement learning. In: Proceedings of the 15th ACM Workshop on Hot Topics in Networks, pp. 50–56. HotNets '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/3005745.3005750
21. Mišić, J., Mišić, V.B., Chang, X., Motlagh, S.G., Ali, M.Z.: Modeling of Bitcoin's blockchain delivery network. IEEE Trans. Netw. Sci. Eng. **7**(3), 1368–1381 (2020). https://doi.org/10.1109/TNSE.2019.2928716
22. Papadis, N., Borst, S., Walid, A., Grissa, M., Tassiulas, L.: Stochastic models and wide-area network measurements for blockchain design and analysis. In: IEEE INFOCOM 2018—IEEE Conference on Computer Communications, pp. 2546–2554. IEEE (2018). https://doi.org/10.1109/INFOCOM.2018.8485982
23. Papadis, N., Tassiulas, L.: Blockchain-based payment channel networks: Challenges and recent advances. IEEE Access **8**, 227596–227609 (2020). https://doi.org/10.1109/ACCESS.2020.3046020
24. Papadis, N., Tassiulas, L.: Payment channel networks: Single-hop scheduling for throughput maximization. In: IEEE INFOCOM 2022—IEEE Conference on Computer Communications, pp. 900–909 (2022). https://doi.org/10.1109/INFOCOM48880.2022.9796862
25. Pickhardt, R., Nowostawski, M.: Imbalance measure and proactive channel rebalancing algorithm for the Lightning Network. In: IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2020, Toronto, ON, Canada, 2–6 May 2020, pp. 1–5. IEEE (2020). https://doi.org/10.1109/ICBC48266.2020.9169456
26. Poon, J., Dryja, T.: The Bitcoin Lightning Network: Scalable off-chain instant payments (2016). https://lightning.network/lightning-network-paper.pdf
27. Russell, R.: Splicing Proposal (2018). http://www.lists.linuxfoundation.org/pipermail/lightning-dev/2018-October/001434.html
28. Togami, W., Nick, K.: PeerSwap: Decentralized P2P LN balancing protocol (2021). http://www.blockstream.com/assets/downloads/2021-11-16-PeerSwap_Announcement.pdf
29. Varma, S.M., Maguluri, S.T.: Throughput optimal routing in blockchain-based payment systems. IEEE Trans. Control. Netw. Syst. **8**(4), 1859–1868 (2021). https://doi.org/10.1109/TCNS.2021.3088799



Game-Theoretic Randomness for Proof-of-Stake

Zhuo Cai(✉) and Amir Goharshady

Department of Computer Science and Engineering, Hong Kong University of Science and Technology (HKUST), Clear Water Bay, Hong Kong SAR, China
zcaiam@connect.ust.hk, goharshady@cse.ust.hk

Abstract. Many protocols in distributed computing rely on a source of randomness, usually called a random beacon, both for their applicability and security. This is especially true for proof-of-stake blockchain protocols in which the next miner or set of miners have to be chosen randomly and each party's likelihood to be selected is in proportion to their stake in the cryptocurrency. The chosen miner is then allowed to add a block to the chain. Current random beacons used in proof-of-stake protocols, such as Ouroboros and Algorand, have two fundamental limitations: Either (i) they rely on pseudorandomness, e.g. assuming that the output of a hash function is uniform, which is a widely used but unproven assumption, or (ii) they generate their randomness using a distributed protocol in which several participants are required to submit random numbers which are then used in the generation of a final random result. However, in this case, there is no guarantee that the numbers provided by the parties are uniformly random and there is no incentive for the parties to honestly generate uniform randomness. Most random beacons have both limitations. In this work, we provide a protocol for distributed generation of randomness. Our protocol does not rely on pseudorandomness at all. Similar to some of the previous approaches, it uses random inputs by different participants to generate a final random result. However, the crucial difference is that we provide a game-theoretic guarantee showing that it is in everyone's best interest to submit uniform random numbers. Hence, our approach is the first to incentivize honest behavior instead of just assuming it. Moreover, the approach is trustless and generates unbiased random numbers. It is also tamper-proof and no party can change the output or affect its distribution. Finally, it is designed with modularity in mind and can be easily plugged into existing distributed protocols such as proof-of-stake blockchains.

Keywords: Distributed randomness · Proof-of-stake · Mechanism design

The research was partially supported by the Hong Kong Research Grants Council ECS Project Number 26208122, the HKUST-Kaisa Joint Research Institute Project Grant HKJRI3A-055 and the HKUST Startup Grant R9272. Z. Cai was supported by the Hong Kong Ph.D. Fellowship Scheme (HKPFS). Authors are ordered alphabetically.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
P. Pardalos et al. (Eds.): MARBLE 2023, 2024.
https://doi.org/10.1007/978-3-031-48731-6_2

1 Introduction

Proof of Work Bitcoin, the first blockchain protocol, was proposed by Satoshi Nakamoto to achieve consensus in a decentralized peer-to-peer electronic payment system [23]. In Bitcoin and many other cryptocurrencies, the miners are selected by a proof-of-work (PoW) mechanism to add blocks of transactions to the public ledger [21], i.e. they have to compete in solving a mathematical puzzle and each miner's chance of adding the next block is proportional to their computational (hash) power. Security guarantees are then proven with the assumption that more than half of computational power is in the hands of honest miners. Proof of work is known to be highly energy-inefficient [2,10] and also prone to centralization due to large mining pools [3]. Currently, the three largest mining pools have more than half of the entire Bitcoin mining power.

Proof of Stake [19] Proof of Stake (PoS) is the main alternative consensus mechanism proposed to replace PoW in blockchain protocols. In a PoS protocol, miners are chosen randomly and each miner's chance of being allowed to add the next block is normally proportional to their stake in the currency. Hence, instead of relying on the assumption that a majority of the computational power on the network is owned by honest participants, the security claims of proof-of-stake protocols rely on the assumption that a majority, or a high percentage, of the stake is owned by honest participants. Despite their differences, all proof-of-stake protocols require a random beacon to randomly select the next miners in an unpredictable manner.

Distributed Randomness A random beacon is an ideal oracle used in a distributed protocol, e.g. a proof-of-stake blockchain, that emits a fresh random number in predetermined intervals. Designing random beacons is an active research topic in the context of distributed and decentralized computation [7,20,26,28,29]. The desired properties of a random beacon are as follows:

- *Bias-resistance*: The output should always be sampled according to a fixed underlying distribution δ, which is usually the uniform distribution. No party should be able to bias the output or change the distribution δ.
- *Unpredictability*: No party should be able to predict the output before it is publicized. Moreover, no party should even be able to have any extra information about the output, other than the fact that it will be sampled from δ.
- *Availability*: Each execution of the beacon must successfully terminate and produce a random value.
- *Verifiability*: Each execution of the beacon should provide a "proof" such that any third party, even if not involved in the random beacon, is able to verify both the output and the fact that the random beacon executed successfully.

Reliable Participants Almost all distributed randomness protocols have several participants and create a random output based on random numbers submitted by participants of the protocol. Usually, the final value is simply defined

by the modular sum of all input values by participants modulo some large number p, i.e. $v := \sum_{i=1}^{n} s_i \pmod{p}$. If the protocol generates only a single random bit, then $p = 2$ and the modular sum is equivalent to the `xor` operation. Using the summation formula above, if the input values of different participants are chosen independently and if at least one of the participants submits a uniform random value in the range $[0, p-1]$, then the final output is also a uniform random value. We call such a participant *reliable*. Note that it is enough to have only one reliable participant for the final output to have the uniform distribution when the values are submitted independently. Therefore, the distributed randomness protocols typically assume that at least one of the participants is reliable. We distinguish between *reliable* and *honest* participants.

Honest Participants An honest participant is a participant who correctly follows the protocol, e.g. submits their random number s_i in time. Distributed randomness protocols often assume and require that a large proportion of participants are honest and obey the communication rules to complete and produce final values. For example, PBFT achieves Byzantine agreement in a partially-synchronous network by requiring that more than two thirds of all participants be honest [8].

Commitment Schemes Using the formula above for random number generation, since the participants cannot broadcast their values in a distributed network in a perfectly simultaneous way, the last participant has an advantage and can dominate the final output. The classical cryptographic primitive used to avoid such a scenario is a commitment scheme. A commitment scheme runs in two phases: a commit phase and a reveal phase. In the commit phase, instead of broadcasting the value s_i directly, each party p_i broadcasts $h(s_i, r_i)$, where h is a cryptographic hash function and r_i is a randomly chosen nonce. In the reveal phase, each party broadcasts the values of s_i and r_i and everyone on the network can verify that the broadcast values have the right hash and thus the party has not changed their choice s_i since the commit phase. However, a commitment scheme does not ensure availability, since malicious parties might only commit but not reveal their values.

PVSS Publicly verifiable secret sharing (PVSS) is a powerful cryptographic tool to ensure the revelation of values s_i even if a number of malicious parties stop participating in the reveal phase of a commitment scheme [27]. PVSS adds a protection layer to traditional secret sharing schemes in the presence of malicious participants. In a PVSS scheme, a dealer is required to provide a non-interactive zero-knowledge proof (NIZK) along with encrypted secret shares $E_i(s_i)$ to guarantee the validity of secret shares. During the reconstruction phase, a participant sends their secret share to other participants along with an NIZK proof to guarantee the correctness of secret share. The NIZK proofs can be verified by any party, including third parties who are not taking part in the PVSS scheme.

RANDAO [1] and EVR [30] RANDAO is a family of smart contracts that produce random numbers. Anyone can participate and submit a random value to contribute to the output. RANDAO uses a commitment scheme. Compared to

general distributed randomness protocols based on distributed networks, RANDAO's smart contracts run on a blockchain with consensus and directly interact with the underlying cryptocurrency. Therefore, RANDAO naturally enjoys the decentralized consensus provided by the blockchain protocol. Besides, economic incentives can be designed to promote honesty. Cheaters who violate the rules are punished economically, e.g. by having their deposits confiscated. On the other hand, honest participants are rewarded by the income generated from providing the random number generation service to external contracts. However, there is no way to ensure bias-resistance and availability. A malicious party might choose not to reveal their value s_i as it might be beneficial to them to bias the output. So, if a party does not reveal values, the whole random number generation process should be repeated, but even this biases the output as a malicious party can choose not to reveal only when the final result is not to their benefit in an external smart contract. Finally, RANDAO does not incentivize reliability and assumes that a reliable party exists, without arguing why. Economically Viable Randomness (EVR) also uses a commit-reveal scheme to generate randomness. It designs a punishment scheme to discourage deviating from the protocol. However, similar to RANDAO, the incentives of EVR only care about whether the values are revealed faithfully. They do not differentiate between a reliable participant who submits a fresh uniformly-sampled random number and an unreliable honest participant who submits a constant number each time while following the rest of the protocol.

VDFs Verifiable delay functions [6] can be used to ensure bias-resistance in distributed randomness protocols. A VDF is a function whose evaluation takes at least some predetermined number of sequential steps, even with many parallel processors. Once the evaluation is complete, it can provide a publicly verifiable proof for the evaluation result, which can also be checked by any third party efficiently.

VRFs Verifiable random functions [14,22] are widely used in PoS blockchain protocols [13,17]. A party can run a VRF locally, producing a pseudo-random output value based on their secret key and random seed. The VRF also outputs a proof of the output that can be verified by anyone with access to the party's public key and random seed. With the use of VRFs, malicious parties cannot predict who the future miners are before the miners announce their identities themselves.

Algorand Algorand [17] is a proof-of-stake blockchain protocol based on Byzantine agreement. The random seed for its VRF is based on the VRF of the previous round. While this guarantees most of the desired properties, a major drawback of this randomness beacon is that the generated numbers are not guaranteed to be uniform.

Ouroboros and Ouroboros Praos Ouroboros [19] was the first provably secure proof-of-stake blockchain protocol. It uses a publicly verifiable secret sharing scheme to generate a fresh random seed for each epoch. However, in this scheme the participants have no incentive to submit a uniform random

value. In other words, there is no incentive to be reliable, but just to be honest. Ouroboros Praos [13] improves over Ouroboros to be provably secure under a semi-synchronous setting. The random seed of Ouroboros Praos is updated every epoch by applying a random oracle hash function to a concatenation of VRF outputs in the previous epoch. Similar to Algorand, the random numbers are not guaranteed to be uniformly random, despite the fact that they are assumed to be uniform in the security analysis.

Our Contribution Our main contributions are as follows:

- First, we design a novel game-theoretic approach for randomness generation. We call this an RIG (Random Integer Generation) game. RIG efficiently produces a uniform random integer from an arbitrarily large interval. Moreover, we show that the only equilibrium in an RIG is for all participants to choose their s_i uniformly at random. In other words, our RIG ensures that the participants are incentivized not only to be honest, but also to be reliable. This will alleviate the problems with the previous approaches and ensure that all desired properties of distributed randomness are attained.
- We show that our RIG approach can be plugged into common randomness generation protocols with ease. In Sect. 4, we design protocols to implement RIG as a random beacon on general proof-of-stake blockchains. We describe RIG protocols based on commitment schemes and VDFs in Sect. 4.1 and RIG protocols based on PVSS in Sect. 4.2.
- In Sect. 5, we discuss how RIG can be deployed with minor changes in particular proof-of-stake protocols. We cover Algorand [17] and Ouroboros Praos [13].

Our protocols are the first to incentivize participants to be reliable and submit uniform random numbers. In comparison, previous distributed randomness protocols using commitment schemes and PVSS assume that there is at least one reliable participant without incentivizing reliability. In other words, they only reward honesty but assume both honesty and reliability. The reliability assumption is unfounded. Several other randomness protocols, including Algorand and Ouroboros Praos, do not depend on random inputs from participants at all, but instead use real-time data on blockchains and cryptographic hash functions to generate pseudo-random numbers. This pseudo-randomness is not guaranteed to be uniform, even though it is standard to assume its uniformity in security analyses. Hence, there is no guarantee that miners get elected with probabilities proportional to their stake.

2 Preliminaries

2.1 Games and Equilibria

Probability Distributions Given a finite set $X = \{x_1, \ldots, x_m\}$, a probability distribution on X is a function $\delta : X \to [0, 1]$ such that $\delta(x_1) + \cdots + \delta(x_m) = 1$. We denote the set of all probability distributions on X by $\Delta(X)$.

One-Shot Games [25] A *one-shot game* with n players is a tuple $G = (S_1, S_2, \ldots, S_n, u_1, u_2, \ldots, u_n)$ where:

- Each S_i is a finite set of *pure strategies* for players i and $S = S_1 \times S_2 \times \cdots \times S_n$ is the set of all *outcomes*; and
- Each u_i is a *utility function* of the form $u_i : S \to \mathbb{R}$.

In a *play*, each player i chooses one strategy $s_i \in S_i$. The choices are simultaneous and independent. Then each player i is paid a *utility* of $u_i(s_1, s_2, \ldots, s_n)$ units.

Mixed Strategies [25] A *mixed strategy* $\sigma_i \in \Delta(S_i)$ for player i is a probability distribution over S_i, that characterizes the probability of playing each pure strategy in S_i. A *mixed strategy profile* is a tuple $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ consisting of one mixed strategy for each player. The *expected utility* $u_i(\sigma)$ of player i in a mixed strategy profile σ is defined as $u_i(\sigma) = \mathbb{E}_{s_i \sim \sigma_i}[u_i(s_1, s_2, \ldots, s_n)]$. Intuitively, in a mixed strategy, the player is not committing to a single pure strategy, but only to the probability of playing each pure strategy.

Nash Equilibria [24] A *Nash Equilibrium* of a game G is a mixed strategy profile σ, such that no player has an incentive to change their mixed strategy σ_i, assuming they are aware of the mixed strategies played by all the other players. Let σ_{-i} be a tuple consisting of all the mixed strategies in σ except σ_i. Formally, σ is a Nash equilibrium if and only if for all $\tilde{\sigma}_i \in \Delta(S_i)$ we have $u_i(\sigma) \geq u_i(\tilde{\sigma}_i, \sigma_{-i})$. A seminal result by Nash is that every finite game G has a Nash equilibrium [24]. Nash equilibria are the central concept of stability and self-enforceability for non-cooperative games [25], especially in the context of game-theoretic analysis of blockchain protocols [9,12,15,16,18], in which each player maximizes their own utility, i.e. when a game is in a Nash equilibrium, no party has an incentive to change their strategy and hence the game remains in the Nash equilibrium.

In distributed randomness generation, especially for proof-of-stake protocols, we aim to have a committee that plays a game whose output is our random number. Since the players/parties are pseudonymous on a blockchain network and only participate using their public keys, we might have multiple accounts in our committee that are actually controlled by the same person or are in an alliance. Therefore, we need a stronger concept of equilibrium that does not assume a lack of cooperation between any pair of players. Thus, we rely on strong and alliance-resistant equilibria as defined below.

Strong Nash Equilibria [4,5] A *strong Nash equilibrium* is a mixed strategy profile in which no group of players have a way to cooperate and change their mixed strategies such that the utility of every member of the group is increased. Formally, σ is a strong Nash equilibrium if for any non-empty set P of players and any strategy profile $\tilde{\sigma}_P$ over P, there exists a player $p \in P$ such that $u_p(\sigma) \geq u_p(\tilde{\sigma}_P, \sigma_{-P})$. In strong equilibria, the assumption is that the players cannot share or transfer their utilities, so a player agrees to a change of strategies in the alliance P if and only if their own utility is strictly increased. However, if the players can share and redistribute utilities, or if they are indeed controlled by the same person, then a group is willing to defect as long as their *total* utility increases, which leads to an even stronger notion of equilibrium:

Alliance-Resistant Nash Equilibria [11] An *alliance-resistant Nash equilibrium* is a mixed strategy profile σ such that for any non-empty set P of players and any strategy profile $\tilde{\sigma}_P$, it holds that $u_P(\sigma) \geq u_P(\tilde{\sigma}_P, \sigma_{-P})$, where u_P is the sum of utilities of all member of P. In our setting, especially in PoS blockchain protocols, alliance-resistant equilibria are the suitable notion to justify stability and self-enforceability, because a person with a large stake is likely to control multiple players in the randomly selected committee and only care about the overall revenue.

2.2 Publicly-Verifiable Secret Sharing

We follow [27] in our description of PVSS. In a PVSS scheme, a dealer D wants to share a secret s with a group of n participants $P_1, P_2, \ldots, P_n$. The goal is to have a (t, n)-threshold scheme, i.e. any subset of t participants can collaborate to recover the secret s, while any smaller subset of participants cannot recover the secret or obtain any information about it. Moreover, anyone on the network, even those who are not participating, should be able to verify that the dealer is acting honestly and following the protocol.

Initialization We assume that a multiplicative group $\mathbb{Z}_q^*$ and two generators g, G of this group are selected using an appropriate public procedure. Here, q is a large prime number and all calculations are done modulo q. Each participant P_i generates a non-zero private key $x_i \in \mathbb{Z}_q^*$ and announces $y_i = G^{x_i}$ as their public key. Suppose the secret to be shared is s, the dealer first chooses a random number r and publishes $U = s + h(G^r)$, where h is a pre-selected cryptographic hash function. The dealer then runs the main protocol below to distribute the shares that can reveal G^r. The main protocol consists of two steps: (1) distribution, and (2) reconstruction, each of which has two substeps.

Distribution This consists of the following:

- *Distribution of the shares.* The dealer picks a random polynomial p of degree at most $t-1$ with coefficients in $\mathbb{Z}_q$ of the form $p(x) = \sum_{j=0}^{t-1} \alpha_j \cdot x^j$. Here, we have $\alpha_0 = G^r$, i.e. the number r is encoded in the first coefficient of the polynomial and every other α_j is a random number from $\mathbb{Z}_q$. The dealer then publishes the following:
  - *Commitment:* $C_j = g^{\alpha_j}$, for $0 \leq j < t$. This ensures that the dealer is committing to the polynomial and cannot change it later.
  - *Encrypted shares:* For each player P_i, the dealer computes and publishes $Y_i = y_i^{p(i)}$, for $1 \leq i < n$. Intuitively, the dealer is taking the value $p(i)$ of the polynomial p at point i and encrypting it using y_i so that only the i-th player can decrypt it. This encrypted value is then published.
  - Proof of correctness: The dealer provides a non-interactive zero-knowledge proof ensuring that the encrypted shares above are valid. See [27] for details.
- *Verification of the shares.* Anyone on the network, be it a player P_i or a non-participant third-party, can verify the proof and encrypted shares provided

by the dealer to ensure that the dealer is acting honestly, i.e. following the protocol above, and not giving out invalid shares.

Reconstruction This step consists of:

- *Decryption of the shares.* Each party P_i knows $Y_i = y_i^{p(i)}$ and their secret key x_i. Recall that $y_i = G^{x_i}$. Hence, the i-th party can compute $Y_i^{1/x_i} = y_i^{p(i)/x_i} = G^{p(i)}$. They publish G^{p_i} along with a non-interactive zero-knowledge proof of its correctness.
- *Pooling the shares.* Any t participants $P_{i_1}, P_{i_2}, \ldots, P_{i_t}$ can compute the G^r by Lagrange interpolation. More specifically, they know t points $(i_j, p(i_j))$ of the polynomial p that is of degree $t-1$. So, they can find the unique polynomial that goes through these points. Note that after all the shares are decrypted, anyone on the network can use t of the shares to compute the polynomial p and then G^r is simply $p(0)$. However, before the decryption of the shares in the reconstruction step, finding G^r requires the collaboration of at least t participants and no set of $t-1$ participants can obtain any information about G^r. Finally, knowing G^r and U, it is easy to find the secret s, i.e. $s = U - h(G^r)$.

A PVSS scheme can be used to generate random numbers. To do so, we use a separate PVSS scheme for each participant P_i. All n PVSS schemes run in parallel. In the i-th scheme, P_i is the dealer and everyone else is a normal participant. P_i first chooses a random number s_i and then performs the protocol above as the dealer. At the end of the process, all the s_i's are revealed by pooling the shares and we can use $v = \sum_{i=1}^{n} s_i$ as our random number. The upside is that no party can avoid revealing their s_i and hence the protocol satisfies availability. The downside is that every set of t parties can unmask everyone else's choices and hence bias the result. Therefore, in random number generation using PVSS we have to assume that there are at most $t-1$ dishonest participants.

2.3 Verifiable Delay Functions

We follow [6] in our treatment of verifiable delay functions. A verifiable delay function (VDF) is a tuple $V = (\mathbf{Setup}, \mathbf{Eval}, \mathbf{Verify})$ parameterized by a security parameter λ and a desired puzzle difficulty t. Suppose our input space is $\mathcal{X}$ and our output space is $\mathcal{Y}$. V is a triplet of algorithms as follows:

- $\mathbf{Setup}(\lambda, \mathbf{t}) \to (ek, vk)$. This function generates an evaluation key ek and a verification key vk in polynomial time with respect to λ.
- $\mathbf{Eval}(ek, x) \to (y, \pi)$ takes an input $x \in \mathcal{X}$ and produces an output $y \in \mathcal{Y}$ and a proof π. **Eval** may use randomness in computing the proof π but not in the computation of y. It must run in parallel time t with $poly(\log(t), \lambda)$ processors.
- $\mathbf{Verify}(vk, x, y, \pi) \to \{\text{Yes, No}\}$ is a deterministic algorithm that verifies the correctness of evaluation in sequential time $poly(\log(t), \lambda)$.

See [6] for more details and desired properties. Intuitively, anyone can evaluate the VDF using the evaluation key. However, this takes a long time, i.e. at least t steps, even when using parallelization. Even if a malicious participant has strong parallel computational power, they cannot evaluate the VDF significantly faster than an ordinary participant that owns only a single processor. However, after the evaluation is done, verifying the result is easy and much faster and anyone can do the verification using the verification key vk.

The use-case of verifiable delay functions in random number generation is to again defend against dishonest participants who do not reveal their choice in a commitment scheme. We can require every participant to provide a VDF whose evaluation is their choice s_i. Then, even if the participant is dishonest and does not reveal their own choice, other participants can evaluate the VDF and obtain the s_i, hence ensuring availability for our random number generation protocol. However, evaluation takes a long time, and hence the choice will not be revealed while in the commit phase.

Note that both PVSS and VDF methods above can be used to ensure availability and defend against dishonest parties who do not reveal their choices. However, they do not incentivize the parties to be reliable and choose their s_i uniformly at random. This is our main contribution in the next section.

3 Random Integer Generation Game (RIG)

We now provide the main component of our approach, i.e. a game to incentivize reliability in random number generation.

3.1 Overview of RIG

RIG Suppose that we have n players and n is even. A *Random Integer Generation game* (RIG) with n players and $m \geq 2$ strategies is a game G in which:

- For every player $i \in \{1, \ldots, n\}$, we have m pure strategies $S_i = \{0, 1, \ldots, m-1\}$;
- We pair the players such that every even player is paired with the previous odd player and every odd player is paired with the next even player. In other words, $pair(2 \cdot k) = 2 \cdot k - 1$ and $pair(2 \cdot k - 1) = 2 \cdot k$.
- At an outcome $s = (s_1, s_2, \ldots, s_n)$ of the game, the payoff of player $i = 2 \cdot k - 1$ is defined as $u_i(s) := f(s_{2k-1}, s_{2k})$ and the payoff of player $j = 2 \cdot k$ is defined as $u_j(s) := -u_i(s) = -f(s_{2k-1}, s_{2k})$, where

$$f(s_{2k-1}, s_{2k}) := \begin{cases} 1 & \text{if } s_{2k-1} - s_{2k} \equiv 0 \pmod{m} \\ -1 & \text{if } s_{2k-1} - s_{2k} \equiv -1 \pmod{m} \\ 0 & \text{otherwise} \end{cases}$$

Essentially, we assume that any adjacent pair of even player and odd player play a zero-sum one-shot game with each other. Their payoffs are independent

of the other $n-2$ players. For each pair $(2 \cdot k - 1, 2 \cdot k)$ of players, this is a zero-sum matrix game with the following payoff matrix $A^{(m)}$ for player $2 \cdot k - 1$:

$$A^{(m)} = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \\ -1 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

3.2 Analysis of Alliance-Resistant Nash Equilibria

Theorem 1. (Alliance-Resistant Nash Equilibrium of an RIG.) *Let G be an RIG game with n players and m strategies, where n is an even number and $m \geq 3$. Let $\bar{\sigma}$ be a mixed strategy profile defined by $\bar{\sigma}_i = (1/m, 1/m, \ldots, 1/m)$ for all i, i.e. the mixed strategy profile in which each player i chooses a strategy in S_i uniformly at random. Then, $\bar{\sigma}$ is the only Nash equilibrium of G. Further, it is also alliance-resistant.*

Proof. First, we prove that $\bar{\sigma}$ is an alliance-resistant Nash Equilibrium. Under the mixed strategy profile, the expected payoff of each one of the players is 0. Let G be a subset of players, then the overall utility of all players in G is $\sum_{i \in G} u_i(\bar{\sigma}_{-G}, \sigma)$ if players in G play another strategy profile σ. Each player i is effectively playing against its adjacent player. If both player i and player $pair(i)$ are in G, then $u_i(\bar{\sigma}_{-G}, \sigma) = -u_{pair(i)}(\bar{\sigma}_{-G}, \sigma)$. The utilities of these two players always sum up to zero, so that the overall utility of G is not influenced by them. Similarly, if both player i and player $pair(i)$ are not in G, they do not influence the overall utility of G either. The only non-trivial part consists of those players in G who play against players outside G. For each such player i, since the player $pair(i)$ plays mixed strategy $\bar{\sigma}_{pair(i)}$, the utility is $u_i = \sigma_i^T \cdot A^{(m)} \cdot \bar{\sigma}_{pair(i)} = \sigma_i^T \cdot (0, 0, \ldots, 0) = 0$. Therefore, the overall utility of G is 0 and changing the strategy has no benefit.

We now prove that $\bar{\sigma}$ is the unique Nash equilibrium of this game. Suppose there is another strategy profile $\tilde{\sigma}$ that is also a Nash equilibrium. Then for any player i, since it is effectively only playing with its adjacent player $j = pair(i)$, it follows that $(\tilde{\sigma}_i, \tilde{\sigma}_j)$ forms a Nash equilibrium for the zero-sum bimatrix game defined by $A^{(m)}$.

Now consider the bimatrix game between player i and player j. Let their utility at Nash equilibrium mixed strategy profile $(\tilde{\sigma}_i, \tilde{\sigma}_j)$ be $(\tilde{u}_i, \tilde{u}_j)$. Since it is a zero-sum matrix game, $\tilde{u}_i + \tilde{u}_j = 0$. Without loss of generality, assume that $\tilde{u}_i \leq \tilde{u}_j$, then $\tilde{u}_i \leq 0$. By the definition of Nash equilibrium, player i cannot increase its utility by changing its strategy $\tilde{\sigma}_i$ to any other strategy σ_i, while player j keeps playing the same strategy $\tilde{\sigma}_j$. This indicates that every coordinate of the vector $A^{(m)} \cdot \tilde{\sigma}_j$ is no more than $\tilde{u}_i$, which is at most 0. Let $\tilde{\sigma}_j = (p_0, p_1, \ldots, p_{m-1})$, then the utility of the player i playing pure strategy k is $\delta_k = p_k - p_{k+1 \pmod m} \leq \tilde{u}_i \leq 0$, for each k in $\{0, 1, \ldots, m-1\}$. However, $\sum_{k=0}^{m-1} \delta_k = \sum_{k=0}^{m-1} p_k - \sum_{k=0}^{m-1} p_k = 0$.

So it must hold that $\tilde{\sigma}_j = (1/m, 1/m, \ldots, 1/m)$ and $\tilde{u}_i = \tilde{u}_j = 0$. Since $\tilde{u}_j = 0 \leq 0$, a similar analysis can show that $\tilde{\sigma}_i = (1/m, 1/m, \ldots, 1/m)$. This proves that $\bar{\sigma}$ is the only Nash equilibrium for the Random Integer Generation game.

The theorem above shows that it is in every player's best interest to play uniformly at random, i.e. choose each pure strategy in S_i with probability exactly $1/m$. Moreover, this equilibrium is self-enforcing even in the presence of alliances. Hence, we can plug this game into a distributed random number generation protocol and give participants rewards that are based on their payoffs in this game. This ensures that every participant is incentivized to provide a uniformly random s_i. As mentioned before, even if one participant is reliable and submits a uniformly random s_i, then the entire result $v = \sum s_i$ of the random number generation protocol is guaranteed to be unbiased. Hence, instead of assuming that a reliable party exists, we incentivize every party to be reliable.

3.3 Dense RIG Bimatrix Game

If the strategy size m is large, which is the case when we aim to generate integers from a large range, then the matrix A would be sparse. If the number of players is much smaller than m, then the probability that one party really receives a non-zero payoff is negligible. Therefore, it is desirable to design a matrix B that is dense and also provides the same unique alliance-resistant equilibrium property as A.

For any integer f such that $1 \leq f \leq m/2$ and $\texttt{gcd}(f, m) = 1$, we can construct a new matrix $B^{(m,f)}$ with dense parameter $2f/m$ such that $B^{(m,f)}_{i,j} = g(j - i)$, where $g(\cdot)$ is defined as:

$$g(l) := \begin{cases} 1 & \text{if } 0 \leq l \leq f - 1 \pmod m \\ -1 & \text{if } m - f \leq l \leq m - 1 \pmod m \\ 0 & \text{otherwise.} \end{cases}$$

For $m = 8$ and $f = 3$, the new payoff matrix $B^{(8,3)}$ is the following:

$$B = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 & 0 & 0 & -1 & -1 \\ -1 & -1 & 1 & 1 & 1 & 0 & 0 & -1 \\ -1 & -1 & -1 & 1 & 1 & 1 & 0 & 0 \\ 0 & -1 & -1 & -1 & 1 & 1 & 1 & 0 \\ 0 & 0 & -1 & -1 & -1 & 1 & 1 & 1 \\ 1 & 0 & 0 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & 0 & 0 & -1 & -1 & -1 & 1 \end{pmatrix}$$

When $f = 1$, the matrix $B^{(m,1)}$ is the same as $A^{(m)}$. We can check that the mixed strategy profile $\bar{\sigma}$ under $B^{(m,f)}$ is also an alliance-resistant Nash equilibrium. To show that it is the only Nash equilibrium, we follow the analysis we did for $A^{(m)}$. Suppose there is another Nash equilibrium $(\tilde{\sigma}_i, \tilde{\sigma}_j)$ between

two adjacent players i and $j(=i+1)$ and $\tilde{u}_i \leq 0$. Let $\tilde{\sigma}_j$ be $(p_0, p_1, \ldots, p_{m-1})$, then every element of $r = B \cdot \tilde{\sigma}_j$ is at most $\tilde{u}_i$, which is non-positive. However, $\sum_{k=0}^{m-1} r_k = \mathbf{1}^T \cdot r = \mathbf{1}^T \cdot B \cdot \tilde{\sigma}_j = \mathbf{0}^T \cdot \tilde{\sigma}_j = 0$, which requires $r = \mathbf{0}$ and $\tilde{u}_i = 0$. By $r = \mathbf{0}$, we have

$$r_s = \sum_{0 \leq l \leq f-1} p_{l+s} - \sum_{m-f \leq l \leq m-1} p_{l+s} = 0$$

for every $s \in \{0, 1, \ldots, m-1\}$. With a slight misuse of notation, assume that $p_{m+t} = p_t$ for any integer t. If we subtract r_{s+1} by r_s, we get

$$p_{s+f} - p_s = p_{s+m} - p_{s+m-f} = p_s - p_{s-f}$$

Let $q(s) = p_s - p_{s-f}$, then $q(s+f) = q(s)$. We also have $q(s+m) = q(s)$. Since $\texttt{gcd}(f, m) = 1$, the function $q(\cdot)$ is constant on integers, from which we can infer that $p_0 = p_1 = \cdots = p_{m-1} = 1/m$. Therefore, $\bar{\sigma}$ is still the only Nash equilibrium.

Important Remark on Parallelization Note that the parallelization of RIG loses the uniqueness property of Nash equilibrium, hence we cannot simply have an RIG game for $m = 2$, use it to generate a random bit, and parallelize it k times to generate k random bits. Instead, we must set $m = 2^k$ and have a single non-parallel RIG game. As an example, consider the simplified case of two players and k bits. If each player only uniformly randomly set their first bit, and then copied the same bit to all other bits, this would also form a Nash equilibrium. However, this Nash equilibrium does not produce a uniform random output. Instead, the output is 0 with 50% probability and $2^k - 1$ with 50% probability. More generally, any $\sigma = (\sigma_1, \sigma_2)$ such that $\sigma_i(j\text{-th bit is } 0) = 1/2$ for all $1 \leq j \leq m$ is a Nash equilibrium. The existence of these undesirable Nash equilibria breaks the guarantee of uniform random distribution over the final output value. Hence, parallelization should not be used and a game on 2^k strategies should be played in the first place.

4 Designing a Random Beacon Based on RIG

In this section, we discuss how to use a single execution of the Random Integer Generation game in a distributed random number generation beacon. The major challenge is to execute the game, in which the parties have to move simultaneously, in a decentralized environment. We propose two schemes to implement the RIG game: (1) using commitment schemes and verifiable delay functions, and (2) using publicly verifiable secret sharing. We assume that the set of players is already fixed. Usually, only a small subset of users (or blockchain miners) are selected from all the users in the system to join a single execution of the game (generation of a random number). The selection rule is determined by the specific application. The design and amount of reward/penalty and deposits are also subject to the specific application. We will address these adjustable configurations in Sect. 4.3. Finally, we focus on the case where our protocol is

used in conjunction with a blockchain protocol, including gossip protocols for announcements.

4.1 Commitment Scheme and VDF Approach

As mentioned above, commitment schemes are already widely used in random number generation in distributed systems. As expected, our approach has two phases: commit and reveal. The execution starts with the commit phase, which lasts for T_{commit} time slots. In a blockchain ecosystem, we can use the block number to keep track of time. After the commit phase ends, the execution enters the reveal phase, which lasts for T_{reveal} time slots. The RIG game is executed when the reveal phase completes.

In the commit phase, each participant p_i broadcasts a signed commit message: $(ss_id, h_i, proof_i)_i$, where ss_id is the session id of the execution and $h_i = hash(s_i|r_i)$ is the commitment. s_i is the value the participant chooses and r_i is a random nonce. $proof_i$ is a publicly verifiable proof that p_i is an eligible participant, applicable when only a subset of selected users are allowed to join the game. A commit message is valid if: (1) the message is properly signed by p_i, (2) the message has a valid proof of participation, (3) there is no other different valid commit message from p_i in the network, and (4) the message is received during the commit phase.

In the reveal phase, each participant p_i broadcasts a signed reveal message: $(ss_id, s_i, r_i)_i$. A reveal message is valid if (1) the message is received during the reveal phase, (2) p_i has exactly one valid commit message, and (3) $hash(s_i|r_i)$ matches the commitment h_i of the participant p_i.

After the reveal phase completes, we can compute the payoffs of the RIG game. We describe in Sect. 4.3 the details of computing results. Assume the outcome of the game is $(s_1, \ldots, s_n)$, where $s_i \in \{0, 1, \ldots, m-1\}$ is the strategy played by each player. We set $v := \sum s_i \pmod m$ and output it as the result of the random number generation protocol.

The value of v can be biased by malicious participants who might choose not to reveal their values/strategies. If a participant does not reveal the values after completing the commit phase correctly, they will lose their deposit. However, the participant might benefit from a biased output of the random beacon in the downstream application, for which they might be willing to cheat even at the cost of losing the deposit. In order to prevent this possibility of adversarial manipulation on the game result, we make use of a verifiable delay function VDF$(\cdot)$ as in [6] and require the participant to provide all necessary parameters for the evaluation of the VDF as part of the commit message. We then check that the provided VDF really evaluates to the strategy s_i of the player and otherwise, simply remove player i from the game. Of course, the VDF evaluation time should be long enough to ensure it cannot be evaluated until the reveal phase is over.

Using this technique, any adversary cannot have any information about the final output by the end of the reveal phase. Therefore, revealing values honestly is always a strictly better strategy than not revealing values for all participants.

The game is then executed when all the values are revealed and all the VDFs are evaluated and it is ensured that cheating participants are excluded.

Note that, even if v conforms to a uniformly random distribution, the output of VDF on v is not guaranteed to be uniformly random. In existing constructions of random beacons that rely on VDF, a hash function is applied to the output of VDF to get random numbers, under the random oracle assumption. However, with the novel RIG game, we can guarantee the delivery of uniformly random output values, if we define the final output as $\tilde{v} = v_1 + \mathrm{VDF}(v_2)$, where v_1 is the higher half bits of v and v_2 is the lower half bits of v. Since v is uniformly random, then v_1 and v_2 are independent uniformly random integers. Whatever distribution $\mathrm{VDF}(v_2)$ has, the sum $\tilde{v}$ is a uniformly random integer.

Finally, note that this approach works as long as at least one of the participants is honest. So, in a proof-of-stake scenario, if we choose n participants for each random number generation, we need to ensure that honest miners have much more than $1/n$ fraction of the stake in the cryptocurrency, so as to ensure that at least one honest participant is chosen with high probability. We also assume that at least one participant is reliable, but this is already incentivized by our game.

4.2 PVSS Approach

The drawback of the commitment scheme in random number generators is the possibility of adversarial manipulation in the reveal phase by not revealing values. We have already seen a solution using VDFs. A publicly verifiable secret sharing scheme solves the same issue differently, i.e. by forcing the opening of commitments, at the cost of increased communication complexity. We follow the PVSS scheme in [27].

An execution of the RIG game in a PVSS scheme consists of three phases: prepare, distribute and reconstruct.

In the prepare phase, all eligible participants inform each other that they are selected. Under a synchronous communication setting, all honest participants can reach a consensus on the list of participants. More specifically, we assume a blockchain protocol that a transaction will be added to a finalized block within known bounded time slots after it is broadcasted. Each participant firstly broadcasts a signed prepare message $(ss_id, proof_i)_i$ to announce its identity along with eligibility proof for the current session of execution. By the end of prepare phase, all prepare messages should be included in the blockchain and are synchronized across all nodes. Suppose the list of participants is $\{P_i\}_{i=1}^{n}$.

In the distribute and reconstruct phases, each participant P_i runs a PVSS scheme to share their value s_i to the other $n-1$participants. This is exactly as described in Sect. 2.2. In the distribute phase, every participant should send valid $(n-1, t)$-threshold secret shares to others along with a proof of commitment and consistency. The shares are publicly verifiable so that a dishonest participant who distributes invalid shares can be discovered and excluded from the game. Hence, by the end of distribute phase, all honest participants release their correct shares and receive correct shares from other honest participants. If a dishonest

participant distributes some invalid shares or does not distribute part of the shares, they will be reported and deleted from the list of participants. As long as the number of dishonest participants is less than t, they cannot decrypt any secret from honest participants in the distribute phase.

In the reconstruct phase, each participant can reveal their value and share the decrypted secret shares they received. If the number of honest participants is at least t, then the pooling mechanism is successful and anyone can find all the secrets from valid secret shares, without the help of any dishonest participant. The dishonest participants cannot mislead honest participants by providing wrong decryption of secret shares in reconstruction, because the decryption for reconstruction also requires a publicly verifiable proof.

Suppose the number of dishonest participants is f. PVSS requires $f < t \leq n - f$. Therefore, we can assume that $n \geq 2 \cdot f + 1$ and let $t = \lceil n/2 \rceil$. In other words, we need to assume that more than half of the participants are honest. In a proof-of-stake use case, the set of participants for each execution are sampled from the population of miners based on their stake. If we want $n \geq 2 \cdot f + 1$ to hold with overwhelming probability, then the dishonest stake ratio should be strictly smaller than $1/2$ and preferably much smaller. Moreover, n should not be too small. In other words, this approach assumes that most of the stake, strictly more than half and preferably much more than half, is owned by honest participants. This is in contrast to the previous approach that only needed more than $1/n$.

4.3 Further Details of the Approach

Participant Selection Rules Proof-of-stake blockchain protocols are important applications of random beacons. To prevent Sybil attacks and enforce proof-of-stake, it is common to sample a small subset of participants based on stake ratios for the random beacon. Verifiable random functions (VRF) [22] are popular for the purpose of selecting participants. VRF requires a random seed, which can be the output of RIG game in the previous round. Similar to the treatment for VDF outputs to ensure uniformly random distribution, we can also use the trick of separating the bits of the random seed v to two parts v_1 and v_2 and using $v_1 + VRF(v_2)$ instead of $VRF(v)$.

Sorting Rules In contrast to RANDAO and many other random number generators, our RIG game is sensitive to the order of participants. The result of the RIG game is not only the output value, which is the sum of all valid values submitted by the participants, but also the payoffs. The honest participants who reveal their values faithfully might receive different rewards/penalties depending on the ordering of participants. As before, we can use the output of the previous RIG round to generate a random ordering for the current round. Finally, the RIG game requires an even number of participants, so if the number of valid participants is odd, we will remove one participant arbitrarily. To make sure this does not have an effect on consensus, we can remove the participant for whom $h($`commit message`$)$ has the largest value.

Design of the Incentives Every participant puts down a deposit d at the same time they send their commit message. The value of d is fixed by the protocol. After collecting all the valid values s_i and ordering of the participants P_i, $i \in [1, n]$, the result has the format $(s, \{P_i, u_i\})$, where u_i is the payoff of participant P_i. The values s, s_i are in $\{0, 1, \ldots, m\}$, where m is a parameter of RIG game, i.e. the number of strategies for each player. The output random number is computed as $s = \sum_{i=1}^{n} s_i \pmod m$. Note that all dishonest players are excluded from the sum. If a player does not release their value or otherwise cheats, then they will be punished by confiscating their deposit and they will not be included in the game. Each honest participant P_i receives a payoff of the form $rw_i = u_i(s_1, \ldots, s_n) + c$. Recall that $u_i(s_1, \ldots, s_n)$ is the payoff of P_i defined by the game matrix, which sums up to 0 among valid participants. The number c is a constant defined by the specific application. Generally, c should be positive and high enough to motivate honest participants to join the RIG game and perform its steps. When we require the participants to use blockchain transactions for communication, c should at least cover the transaction fees. The deposit amount d should also be larger than any reward that a participant can possibly obtain in the game in order to discourage dishonest behavior.

4.4 Assumptions and Limits to Applicability

Network Assumptions The most important assumptions are the network assumptions. Our RIG game relies on a synchronous communication network. All real-world blockchain networks use the internet and are effectively synchronous.

δ-Synchrony A broadcast network is δ-synchronous if a message broadcasted by some user at time t will be received by all other users by time $t + \delta$.

When applied to blockchains, blockchain consensus protocols guarantee public ledgers with different levels of synchrony. In this paper, we rely on blockchain consensus protocols to achieve a synchronized view of RIG execution. In detail, we require Δ-synchrony for blockchains:

Δ-Synchronous blockchains A blockchain is Δ-synchronous if any valid transaction broadcasted by some user at time t will become part of the stable chain at time $t + \Delta$ in the view of all honest nodes.

We assume that Δ is known to all nodes and use it to design the duration of commitment scheme approach and PVSS approach. If the PVSS approach is implemented using off-chain communication, then we can design durations in terms of δ. In any case, the approach will not work if the network/blockchain is not synchronous or if the time limits are too tight and messages are not guaranteed to be delivered before the beginning of the next phase.

Rationality Assumption We proved that any rational party or parties, i.e. a party/parties interested only in maximizing their own payoff, would play uniformly at random in an RIG game and would therefore be reliable. This is because playing uniformly at random is the only alliance-resistant equilibrium in the game. Moreover, the uniformity of the output random number depends

on having at least one reliable player. Therefore, we must assume that at least one player is rational and our approach would not work if none of the players are rational. However, this case is unlikely to happen in practice as we would normally expect all parties to be rational.

Computation and Communication Complexity Our RIG can be implemented using a commit-reveal scheme or a PVSS scheme, which is typical in random number generation protocols and other components of distributed protocols. Depending on the assumptions such as availability of reliable communication channels, the complexity might be different. Overall, we claim that the computation and communication complexity of RIG is better or comparable with existing efficient random number generation protocols and imposes negligible overhead in applications under reasonable assumptions, as exemplified in Sect. 5.

5 RIG in Proof of Stake Protocols

We now show how our RIG random beacon can supplant standard PoS protocols. In general, the RIG random beacon, be it implemented by the commitment scheme approach or the PVSS approach, is applicable to any PoS protocol that requires an evolving random seed to select miners. Overall, using our RIG as the source of randomness only introduces negligible overhead in terms of transaction throughput.

If we use the RIG random beacon for generating the random seed in a proof-of-stake protocol, a single execution of the RIG game updates the random seed once. Usually, a single execution spans an epoch, which consists of multiple slots where the same random seed is repeatedly used. The blockchain protocol is modified to consecutively run the RIG random beacon to update the random seed in every epoch. The participants of RIG random beacon of each epoch are randomly selected, e.g. based on the RIG result of the previous epoch. Note that our approach can also be applied for every block, instead of every epoch, but this would require more communication complexity.

5.1 RIG in Ouroboros Praos

Ouroboros Praos is the second proof-of-stake protocol in the Ouroboros family and the underlying protocol of Cardano cryptocurrency [13]. The selection rule for random seed generation participants in Ouroboros Praos is based on a VRF. The random beacon concatenates the VRF output of the participants and applies a random oracle hash function on the concatenated output. Each participant is also a miner and announces their VRF output along with their new block. The generated random seed is used in the next epoch, which consists of a number of slots. The protocol waits for enough slots until the seed generation is synchronized among all participants for the next epoch.

We can substitute the random beacon of Ouroboros Praos with our RIG. This is feasible because Ouroboros Praos assumes >50% honesty and suitable

synchronous networks. If we use the commitment scheme approach, we can reuse the VRF selection rule and epoch/slot timing system. The major difference is that the execution of RIG consists of two phases of communication. Therefore, it requires $T_{reveal} + T_{wait}$ more slots within an epoch to reach a consensus on the result of RIG. Besides, we improve the VRF selection using the split randomness trick to ensure uniformly random sampling, instead of pseudo-random sampling. In Cardano, 1 epoch lasts for 5 d, and the transaction confirmation time is 20 min. When using the commitment scheme, we require more time (<= 3× transaction confirmation time) within an epoch for the extra reveal phase and VDF computation time to reach a consensus on the result of RIG, which is negligible.

5.2 RIG in Algorand

It is also feasible to use RIG in Algorand [17]. Algorand executes a byzantine agreement protocol to achieve no-fork and one-minute-liveness properties, which guarantee a one-minute-synchronous blockchain for RIG random beacon. Algorand assumes that honest users hold more than two-thirds of the total stake, which satisfies the requirement of the PVSS implementation of RIG.

Algorand requires an evolving random seed for VRF-based selection of miners and Byzantine agreement protocols. The random seed is updated every R rounds by applying the VRF of the current miner to the previous random seed and the current epoch number. This is again pseudo-random and the output might deviate from uniform distribution. We can use our RIG random beacon to generate the random seeds for Algorand. If we use on-chain communication in the PVSS approach, then we have to select the participants of RIG separately from the committee for each single round, because we want the participants of RIG to be active for multiple slots ($T_{prepare} + T_{distribute} + T_{reconstruct}$ slots) without obstructing the growth of blockchain. Each single execution of the RIG spans the duration of an epoch, which consists of R rounds. Algorand reaches consensus within 1 round, and updates the random seed once every 1000 rounds, which is sufficient for a PVSS execution. Moreover, assuming that $R = 1000$ and 100 participants join the RIG game, using RIG decreases the transaction-per-second by less than 1%.

The execution of the RIG random beacon is parallel to other parts of Algorand. While other parts require an evolving random seed from the RIG random beacon, they can reuse the previous random seed until a new seed is computed and synchronized. Compared to the random seed updating procedure in Algorand, our RIG random beacon is bias-resistant and rules out the possibility of adversarial manipulation on the random seed assuming the selected participants satisfy the assumption of honest majority.

6 Conclusion

We presented a game-theoretic beacon for distributed random number generation. We showed that our approach is bias-resistant, unpredictable, available,

verifiable and incentivizes every participant to be reliable, i.e. faithfully provide a uniform random input. Even if only one of the participants is rational and thus reliable, the output number is guaranteed to be unbiased. Additionally, our approach does not use pseudo-randomness at any point and instead only relies on well-incentivized game-theoretic randomness. Finally, even though the approach is general and not limited to blockchain use cases, we showed that one can easily augment common proof-of-stake protocols to include our randomness beacon for the task of selecting their miners. This ensures that proof-of-stake protocols choose the miners fairly, i.e. exactly in proportion to their stake.

References

1. RANDAO: A DAO working as RNG of Ethereum (2019). www.github.com/randao/randao
2. Cambridge bitcoin electricity consumption index (2022). www.ccaf.io/cbeci/index
3. Arnosti, N., Weinberg, S.M.: Bitcoin: a natural oligopoly. Manag. Sci. 4755–4771 (2022)
4. Aumann, R.J.: 16. Acceptable Points in General Cooperative n-Person Games, pp. 287–324. Princeton University Press (2016)
5. Bernheim, B., Peleg, B., Whinston, M.D.: Coalition-proof nash equilibria i. concepts. J. Econ. Theory 1–12 (1987)
6. Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: CRYPTO, pp. 757–788 (2018)
7. Cai, Z., Goharshady, A.: Trustless and bias-resistant game-theoretic distributed randomness. In: IEEE ICBC (2023)
8. Castro, M., Liskov, B.: Practical byzantine fault tolerance. In: OSDI, pp. 173–186 (1999)
9. Chatterjee, K., Goharshady, A.K., Ibsen-Jensen, R., Velner, Y.: Ergodic mean-payoff games for the analysis of attacks in crypto-currencies. In: CONCUR (2018)
10. Chatterjee, K., Goharshady, A.K., Pourdamghani, A.: Hybrid mining: exploiting blockchain's computational power for distributed problem solving. In: SAC, pp. 374–381 (2019)
11. Chatterjee, K., Goharshady, A.K., Pourdamghani, A.: Probabilistic smart contracts: Secure randomness on the blockchain. In: IEEE ICBC, pp. 403–412 (2019)
12. Chatterjee, K., Goharshady, A.K., Velner, Y.: Quantitative analysis of smart contracts. In: ESOP, pp. 739–767 (2018)
13. David, B., Gaži, P., Kiayias, A., Russell, A.: Ouroboros praos: an adaptively-secure, semi-synchronous proof-of-stake blockchain. In: CRYPTO, pp. 66–98 (2018)
14. Dodis, Y., Yampolskiy, A.: A verifiable random function with short proofs and keys. In: PKC, pp. 416–431 (2005)
15. Farokhnia, S., Goharshady, A.: Alleviating high gas costs by secure and trustless off-chain execution of smart contracts. In: SAC (2023)
16. Farokhnia, S., Goharshady, A.: Reducing the gas usage of Ethereum smart contracts without a sidechain. In: IEEE ICBC (2023)
17. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: scaling byzantine agreements for cryptocurrencies. In: SOSP, pp. 51–68 (2017)
18. Goharshady, A.K.: Irrationality, extortion, or trusted third-parties: why it is impossible to buy and sell physical goods securely on the blockchain. In: IEEE Blockchain, pp. 73–81 (2021)

19. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. In: CRYPTO, pp. 357–388 (2017)
20. Krasnoselskii, M., Melnikov, G., Yanovich, Y.: No-dealer: byzantine fault-tolerant random number generator. In: INFOCOM, pp. 568–573 (2020)
21. Meybodi, M.A., Goharshady, A.K., Hooshmandasl, M.R., Shakiba, A.: Optimal mining: maximizing bitcoin miners' revenues from transaction fees. In: IEEE Blockchain, pp. 266–273 (2022)
22. Micali, S., Rabin, M.O., Vadhan, S.P.: Verifiable random functions. In: FOCS, pp. 120–130 (1999)
23. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Decentralized Business Review, p. 21260 (2008)
24. Nash, J.: Non-cooperative games. Ann. Math. 286–95 (1951)
25. Roughgarden, T., Nisan, N.: Algorithmic Game Theory. Cambridge University Press (2007)
26. Schindler, P., Judmayer, A., Stifter, N., Weippl, E.R.: Hydrand: efficient continuous distributed randomness. In: SP, pp. 73–89 (2020)
27. Schoenmakers, B.: A simple publicly verifiable secret sharing scheme and its application to electronic voting. In: CRYPTO, pp. 148–164 (1999)
28. Syta, E., Jovanovic, P., Kokoris-Kogias, E., Gailly, N., Gasser, L., Khoffi, I., Fischer, M.J., Ford, B.: Scalable bias-resistant distributed randomness. In: SP, pp. 444–460 (2017)
29. Wang, G., Nixon, M.: Randchain: practical scalable decentralized randomness attested by blockchain. In: IEEE Blockchain, pp. 442–449 (2020)
30. Yakira, D., Asayag, A., Grayevsky, I., Keidar, I.: Economically viable randomness (2020). arXiv:2007.03531



Incentive Schemes for Rollup Validators

Akaki Mamageishvili(✉) and Edward W. Felten

Offchain Labs, Inc., Princeton, USA
amamageishvili@offchainlabs.com

Abstract. We design and analyze attention games that incentivize validators to check computation results. We show that no pure strategy Nash equilibrium of the game without outside parties exists by a simple argument. We then proceed to calculate the security of the system in the mixed Nash equilibrium, as a function of the number of validators and their stake sizes. Our results provide lower and upper bounds on the optimal number of validators. More concretely, a minimal feasible number of validators minimizes the probability of failure. The framework also allows to calculate optimum stake sizes, depending on the target function. In the end, we discuss optimal design of rewards by the protocol for validators.

1 Introduction

Decentralized systems, such as blockchains, rely on validators to compute and verify the state of the system. Validators are rational players that provide a service when the right incentives are in place: they are rewarded for correct behavior and punished for misbehavior. Blockchains have successfully implemented these incentives. The validators stake some amount of (blockchain native) tokens and earn rewards if they provide validation service. If they misbehave, there are two ways they get punished. First, directly, by slashing stakes. Second, indirectly: if the system loses security, it causes the validator stake to be devalued or even stolen.

There are still many research directions to explore. We study a particular setting in this paper, specifically, a setting where a single honest and active validator is necessary for the security of a system, as for example described in [5]. We ask a question exactly how many validators are required for a high enough security level of the system, and what their stake sizes should be. Our model and insights are general enough to be applied to a broad class of decentralized systems, but they are best suited to optimistic rollup protocols. To scale the base layer protocol, computationally heavy execution is delegated from the base layer to layer two protocols, called rollup protocols. In case of optimistic rollups, only claims on intermediate states are made on the base layer. This is where active validation becomes important. If one party, typically one of the validators, wrongly claims a state of the system, other validators are supposed to challenge and prove that the claim is wrong. A single honest validator is enough to disprove a wrong claim, and therefore, guarantee the security of the protocol.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
P. Pardalos et al. (Eds.): MARBLE 2023, 2024.
https://doi.org/10.1007/978-3-031-48731-6_3

In the first part of the paper, we study a game where validators obtain rewards only when they find false claims made by other (malicious) validators. In the context of optimistic rollups, only one honest validator which checks computation results is enough to detect and prevent malicious attackers' efforts. The malicious asserter can make a false claim that allows all assets at the rollup to be stolen if all validators fail to challenge it. This is why long periods for a challenge are given. Successful attacks are subtler on the base blockchains, as they also devalue stolen assets if they are in the native tokens, however, in the rollups the deployed assets are not native and therefore, not devalued on the base layer.

We first consider a game in which the behavior of players is completely based on economic incentives. In particular, there are no social norms and the payoffs of the players are not affected by outside parties, e.g., the system designer does not inject money into the system. The reward a validator might receive is paid by another validator that made a false claim about the system state. This way the game is similar to the so-called multi-player *zero-sum game*. The difference with the zero-sum game, however, is that if the malicious validator succeeds with a false claim, it may obtain a much higher payoff than what the rest of the validators lose.

In this model, we argue that a pure strategy equilibrium does not exist, and therefore, we focus on mixed strategy equilibrium solution concept. In this solution, we calculate the probability of the system failure, which should serve as an upper bound on the system security expected if there are only rational players. In reality, we expect the system to be more secure because there are altruistic players and some players may also care about their reputation.

In the second part, we allow rewards to validators without detecting false claims. To achieve this, validators are required to post their checking results on-chain with some probability. To be able to figure out if posting is needed, validators need to compute the state of the system fully. If the probability of posting is high enough, validators are incentivized to do computation all the time, hence, pure strategy equilibrium is obtained. They are rewarded when they correctly post about the state of the world, the amount which compensates for their past checking efforts as well.

Related Literature

The question of correctly incentivizing validators in blockchain settings is not recent, see [7], where the problem is referred to as the *verifier's dilemma*. In parallel research, [6] studies the security of optimistic rollups in a very similar model as ours. The differences are the absence of deposits of the validators, equivalent to our definition of a silent validator and focusing on 2 players, a representative attacker, an *Asserter* in our case, and a representative validator. A closely related paper to ours is [8]. The difference between our approaches is that in [8] validators (data providers) are not explicitly rewarded for their service. Instead, the authors look at only punishing strategies to incentivize the right behavior. Also, they consider pure strategy Nash equilibria solutions. [3] studies a similar question of how to design a bug-searching committee. The paper

studies the question of the optimal number of searchers and reward policies, given a fixed budget. [2] studies blockchain security through the lenses of game theory, in particular, how to design a sharing of validation rewards. However, in this model, the costs of validation differ across validators. Another related paper is [4], where costs of validation differ, and the authors look at the problem of delegating validation, a common way of earning through rewards. Game theory and security literature have a big intersection. For research relating these two in the context of multi-party computation settings as well, see [1].

2 Model

We first consider the simplest case of two validators, which we sometimes refer to as players. One of them is playing the role of the Asserter and the other playing the role of the Checker. This setting corresponds to the system with two validators. Most of the parameters are defined in the game with two players, denoted by $\mathcal{G}_2$ and later used for a more general setting with n players. The roles of players are specified in the following:

- The Asserter makes a claim about the state of a system that is either true or false. The state of the system is a result of a computation of a state transition function, that is fed a stream of incoming transactions.
- The Checker can check the claim made by the Asserter, at some computational cost. The Checker has the option to challenge the claim.

The Asserter's action set consists of false and true claims. The Checker's action set consists of checking and not checking.

The following is a list of the parameters of the game that define the final utilities of the players.

- C: the cost of checking. Here we refer to the cost of checking each claim about the state of the system. Claims arrive at a regular rate. We ignore the sunk costs of setting up the validator node. After such a node is set up, the validator that runs a node pays only server and maintenance costs. Node software that does a check of the claim is typically available for validators of the rollup systems. Therefore, it is safe to assume that they incur homogenous costs of checking.
- R: the deposit of the Asserter. In case of false claim discovery, R is also the Checker's reward. In practice, only a fraction of the reward is given to a Checker and the rest is burnt. This is done in order to avoid situations where both validators are controlled by the same malicious player. In such a case this malicious player does not lose any deposit by rising a false assertion and losing the challenge to itself.[1] However, in the analysis, we assume that validators are not controlled by the same player and the mechanism is budget balanced: what one player loses goes to another player. The analysis does not change and results are the same asymptotically (and qualitatively) if we assume burning some fraction of the Asserter's deposit.

[1] Such delay is an attack on its own.

- L: the Checker's loss if cheating goes undetected. Note that L does not have to be equal to R. It can be a stake validators have locked for being a validator. However, by the design of a challenge mechanism, L is at most R.
- U: is the Asserter's gain if cheating goes undetected. It can be thought of as the full assets locked in the protocol, sometimes referred to as *total value locked (TVL)*. This interpretation of the value is most suited to rollup protocols.

strategy	false (π)	true ($1-\pi$)
check (α)	$R-C, -R$	$-C, 0$
don't check ($1-\alpha$)	$-L$, U	0,0

The table above gives a game $\mathcal{G}_2$ in a bimatrix format. The first number is the utility (payoff) of the Checker, the second number is the utility of the Asserter. The Checker is sometimes referred to as a row player and the Asserter – a column player.

We make a mild assumption on the parameters:

Assumption 1. Assume that the Checker does not have a dominant strategy, that is, $R - C > -L$, equivalent to $C < R + L$.

The condition is intuitive: if the cost of checking is too high, the Checker never checks. A common goal is to minimize the probability of a system failure, denoted by $F(C, L, R, U)$. That is, the probability that a false claim is introduced by the Asserter and the Checker does not check it. Note that the Asserter does not have a dominant strategy, given the assumption. Since the Checker does not have a dominant strategy, we can only have a totally mixed equilibrium game. A similar observation is made in [6]. However, in the totally mixed equilibrium solution of [6], in our case, because of the assumptions, the probability of a successful attack is very low, but the damage is huge, while in the case of [6], both probability and damage are moderate. We assume that if an attack is detected, then the Asserter gets nothing, while in [6] gives the same π payoff in both cases. In the case of detection, the payoff is negative because of a larger deposit slashed, while in our case it is just negated deposit, $-R$. Second, we assume that in case of a successful attack, the attacker can in principle steal all assets, while [6] assumes that the π is a moderate value.

We compute probabilities in the mixed equilibrium solution in the following. Consider mixed strategies for both players. For the Asserter, strategies can be characterized by the probability π that the chosen action (claim) is false. With probability $1 - \pi$, the claim is true. For the Checker, the mixed strategy is characterized by the probability α that the Checker checks. With probability $1 - \alpha$, the Checker does not check.

Proposition 1. *The probability of failure is increasing in the cost of checking C and decreases in U.*

Proof. We can now calculate equilibrium probabilities α and π, using indifference conditions. The indifference condition for the Checker is that the expected utility of playing "check" is equal to the expected utility of playing "don't check". That is,

$$\pi(R-C)+(1-\pi)(-C)=\pi(-L), \tag{1}$$

equivalent to

$$\pi=\frac{C}{R+L}. \tag{2}$$

The indifference condition for the Asserter is that the expected utility of playing "false" is equal to the expected utility of playing "true". That is,

$$0\alpha+0(1-\alpha)=(-R)\alpha+U(1-\alpha), \tag{3}$$

equivalent to

$$\alpha=\frac{U}{R+U}. \tag{4}$$

By plugging in the equilibrium values of π and α, we obtain:

$$F(C,L,R,U):=\pi(1-\alpha)=\frac{CR}{(R+L)(R+U)}. \tag{5}$$

It is easy to see that $F(C,L,R,U)$ is decreasing in increasing U.

If the rollup has a higher value, it is less likely to fail in the equilibrium. The explanation is simple: a higher value of the rollup protocol makes it more attractive for a malicious Asserter to try to make a false claim to transfer all value to itself, but this on the other hand gives more motivation to the Checker to check, as it is earning on finding a false claim. In light of this, another value of interest is the expected loss of the system $F(C,L,R,U)U$. This value is increasing in U, and converges to $\frac{CR}{R+L}$ as U tends to infinity.

Note that, by (4), α is decreasing in increasing R. This sounds counterintuitive – a larger reward for the checker discovering a false claim makes it less likely that the checker will check for a false claim. Even though higher R should increase the incentive of the Checker to check, and all else being equal it does, it also decreases the probability π that the Asserter introduces a false claim, which on its own decreases the incentive for the Checker to check. That is, the recommendation is that increasing R is not the solution to maximize checking probability. Consider a derivative of $F(C,L,R,U)$ as a function of R.

$$\frac{dF}{dR}=\frac{C(R+L)(R+U)-CR(R+L+R+U)}{(R+L)^2(R+U)^2}=\frac{CUL-CR^2}{(R+U)^2(R+L)^2}. \tag{6}$$

Solving $\frac{dF}{dR}=0$ gives $R=\sqrt{UL}$. Therefore, when $R<R^*:=\sqrt{UL}$, LHS of (6) is positive and when $R>R^*$, then it is negative. That is, the probability is decreasing in R above R^*, and should be taken as high as possible.

Note a few observations on F, from the formula (5). First, F is minimized at $R = 0$. It implies that a false claim comes for free, therefore, the Asserter tries a false claim all the time, and the Checker checks all the time, as the cost of checking is less than the punishment for not checking: $C < L$. This would be a desirable solution, but each false claim delays finality, and therefore, harms the system. Therefore, in optimizing the parameter sets, we also care about π, which is maximized by taking $R = 0$. One immediate takeaway from the (2) formula is that if we want to minimize π, we need to increase L and R.

For decreasing π, we need to increase L, equivalent to disincentivizing the Checker to stay idle, causing the Asserter to introduce a false claim less often, and to decrease R, equivalent to incentivizing the Asserter to introduce the false claim more often and, therefore, causing the Checker to check more often.

Note that $\alpha = 1$ for any R is achievable if the Asserter commits to introduce the false claim with a certain minimum probability: $\pi > \frac{C}{R+L}$. However, this can not be sustained in equilibrium: the Checker always checks as it has positive utility, while the Asserter has strictly negative utility: $-\pi \cdot R$. It is not rational for the Asserter. It can only be supported as a solution if, for example, the protocol designer plays the role of the Asserter and posts wrong claims with a probability more than the bound $\frac{C}{R+L}$.

One potential goal a system designer can have is to optimize social welfare, in which the costs of validating and a fraction of the stakes are subtracted from the success probability times TVL. The first cost is obvious—the cost of (duplicate) checking is lost for the validator. The second, opportunity cost, is incurred by the validators by staking their assets in the validation system instead of earning interest outside. The system designer has to minimize the following target function:

$$M := fU\pi + U\pi(1-\alpha) + \alpha C + r(L+R), \tag{7}$$

where f denotes the relative loss of the system when there is a delay and r is a potential return on investment outside the system. f is typically assumed to be a low number, say 10^{-2}. Plugging in the values for π and α gives an equivalent equation to (7):

$$M = f\frac{C}{R+L} + \frac{C}{R+L}\frac{R}{R+U} = \frac{C}{R+L}\left(f + \frac{R}{R+U}\right) + \frac{U}{U+R}C + r(L+R).$$

The optimum values of R and L minimizing M can be computed by taking partial derivatives of M with respect to R and L.

2.1 Extension to $n+1$ Validators

In this section, we assume that there are $n+1$ validators. One of the validators is an Asserter, in each round. We want to incentivize the validators to check claims often enough. In each round, the Asserter makes a claim, and validators can check it. If they check and find the false claim, they do not get slashed if they post a challenge to the false claim. If they check and find out that the claim

is true, they do not need to post anything. That is, not posting anything can mean two things: the validator checked and found out the claim is true, or the validator did not check and that is why there is no post. There are two ways to implement the payoff to the players in the protocol:

1. Simultaneous: every validator posts, if they want to post, at the end of a predefined time interval. This approach is simpler to analyze, and for the players, it is simpler to make a decision.
2. Sequential: The validators see what other validators have done so far. If nobody posts anything this may motivate them not to post anything, but that increases the chances that someone will post a fraud-proof at the last second and slash the silent validators.

For simplicity, we focus on the simultaneous model in this paper. That is, validators see only at the end of the round how many posted checks. Having homogenous costs of checking among validators is a natural assumption in the setting of rollups, as there is available software for running a validator node and standard hardware requirements.

If no validator detects a false claim, the Asserter proceeds with the false claim, and all validators are punished by losing all their deposited stake – L – and the Asserter can steal all value on the chain, giving it payoff U.

We consider a fully mixed symmetric equilibrium of this game. Similarly to the case with two players, the probability that each validator checks is denoted by α, and the probability the Asserter claims a false claim is denoted by π. The timeline of the events is the following:

- If m out of n validators find a false claim and post about it, they are paid equally: $\frac{R}{m}$.
- the other $n - m$ validators are slashed s_w, which we assume to be (much) smaller than L.

The probability that at least one out of n validators will check is equal to

$$P_{s,\alpha}(n) := 1 - (1 - \alpha)^n.$$

Note that in this definition α is an independent parameter, however, in the equilibrium it depends on n. Similarly to (1), we derive the indifference condition for the validator. Shortly, it is EU[check] = EU[don't check], where EU[X] stands for expected utility from taking an action X. The condition can be translated as:

$$\pi \left(\sum_{i=0}^{n-1} \binom{n-1}{i} \alpha^i (1-\alpha)^{n-1-i} \frac{R}{i+1} \right) + (1-\pi)0 - C = \tag{8}$$

$$\pi(P_{s,\alpha}(n-1)(-s_w) + (1 - P_{s,\alpha}(n-1))(-L)) + (1-\pi)0. \tag{9}$$

The first summand on the left-hand side (LHS) represents the product of the probability that the claim is false, the probability that i other validators check

and (expected) rewards $R/(i+1)$, as there are $i+1$ validators finding the false claim.

The second summand on the LHS is a product of the probability that the claim is true with 0, while the last represents the minus cost of checking.

The first summand of the right-hand side (RHS) represents the product of the probability that the claim is false, the product of the probability that someone else checks with $-s_w$.

The second summand of RHS is a product of the probability that nobody checks with $-L$, while the third summand is a product of the probability that the claim is true with 0.

The indifference condition can be further simplified to:

$$\pi\left(\sum_{i=0}^{n-1}\binom{n-1}{i}\alpha^i(1-\alpha)^{n-1-i}\frac{R}{i+1}\right)-C=$$
$$\pi(P_{s,\alpha}(n-1)(-s_w)+(1-P_{s,\alpha}(n-1))(-L)). \tag{10}$$

First, the following claim, obtained in [3], holds.

Lemma 1. *For $n\in\mathbb{N}$ and $x\neq 0$,*

$$\sum_{k=0}^{n}\binom{n}{k}\frac{x^k y^{n-k}}{k+1}=\frac{1}{n+1}\frac{(x+y)^{n+1}-y^{n+1}}{x}. \tag{11}$$

The lemma implies simplification:

$$\sum_{i=0}^{n-1}\binom{n-1}{i}\alpha^i(1-\alpha)^{n-1-i}\frac{R}{i+1}=R\frac{1-(1-\alpha)^n}{n\alpha}. \tag{12}$$

Similarly to (3), the indifference condition of the Asserter is EU[false claim] = EU[true claim]. The LHS is the sum of the product of the probability that some validator checks with $-R$ and the product of the probability that no validator checks with U. The RHS is equal to 0. The condition simplifies to:

$$(1-(1-\alpha)^n)R=(1-\alpha)^n U. \tag{13}$$

From the condition, we obtain a solution for α:

$$\alpha=1-\left(\frac{R}{R+U}\right)^{1/n}.$$

RHS of (12) further simplifies to:

$$R\frac{1-(1-\alpha)^n}{n\alpha}=\frac{UR}{(R+U)n\alpha}.$$

Plugging in α in the equation (10) and taking into account (12) derives π:

$$\pi = \frac{C}{\frac{UR}{(U+R)n\alpha} + P_{s,\alpha}(n-1)s_w + (1 - P_{s,\alpha}(n-1))L}. \tag{14}$$

The main value of interest as in the case with 2 players is $P_{s,\alpha}(n)$. It is obtained by solving the Asserter's indifference condition (13) and is equal to $\frac{U}{R+U}$. That is, it does not depend on the number of validators n. The second value of interest is π, as $\pi(1 - P_{s,\alpha}(n))$ is the probability that the false claim will go through unnoticed. Note that π is decreasing in increasing s_w and L. That is, for decreasing the probability that the Asserter is introducing the false claim, and therefore, decreasing the total probability the false claim goes through unnoticed, we need to increase the slashed stakes of the validators. All else being equal, we obtain the following result:

Proposition 2. *π is increasing with increasing n.*

This result helps to find out the optimum number of validators, in particular, it suggests to rollup systems that n should be as low as possible. On the other hand, $n > 1$ might still be needed because, for example, some validators are not online for some time.

Suppose t validators go offline for a technical reason. We calculate the probability that the system still functions in equilibrium. It is equal to $P_{s,\alpha}(n - t)$. In the following, we give a numerical example.

Example 1. In this example, we consider realistic values of parameters. Suppose C is normalized to 1 (dollar), which is a reasonable approximation of one round computation costs, $U = 10^9$ corresponding to the TVL, $R = 10^6$ corresponding to a stake that an asserting validator needs to commit and $L = 10^5$ corresponding to a stake an active validator needs to commit. The probability of failure is minimized when $n = 1$ and it is equal to $\frac{10^6}{(10^6+10^5)(10^9+10^6)} \approx 10^{-9}$. Now assume that $n = 12$. Then the probability each validator checks, α, is approximately equal to 0.4377, and the probability of a false claim is approximately equal to $3.448 \cdot 10^{-6}$ and the probability of failure equals to $3.445 \cdot 10^{-6}$. When $t = 2$, the probability of failure equals to $0.343 \cdot 10^{-6}$. The driving force of these (good) results is that π is very low. The other multiplier, the probability that one of the validators will check, has a lower effect on the result. The following table shows approximate values of α and π for $n \leq 12$. Note that the probability all validators will fail to catch a false claim, in that case, is $(1 - \alpha)^n = \frac{R}{U+R}$, independent of n, and approximately equals to 10^{-3}.

Table 1. Equilibrium probabilities as a function of n.

n	1	2	3	4	5	6	7	8	9	10	11	12
α	0.999	0.968	0.900	0.822	0.748	0.683	0.627	0.578	0.535	0.498	0.466	0.437
π	9.1e-7	1.9e-6	2.7e-6	3.3e-6	3.7e-6	4.1e-6	4.4e-6	4.6e-6	4.8e-6	5e-6	5.1e-6	5.3e-6

2.2 Silent Validators

In this section, we assume the existence of "silent" validators. They do not stake anything, unlike (active) validators considered so far, but can access the base layer contract after each claim and challenge the (false) claim of the Asserter, in case active validators did not do so.[2] A successful claim by a silent validator allows it to collect all stakes – nL and the Asserter's deposit R. This gives more incentive to the staked validators to check.

To get an intuition, we start with the smallest instance. Assume that there is one active and one silent validator. The indifference condition of the active validator stays the same as in the case without silent validators, as the active validator loses its deposit if the Asserter's claim is false. The indifference condition of the silent validator, on the other hand, is:

$$\pi(1-\alpha)R = C. \tag{15}$$

The expected gains for the silent validator is $(1-\alpha)R$, as the active validator finds the false claim with probability α.

Plugging in $\pi = \frac{C}{R+L}$ in (15) gives a contradiction, the LHS is always lower than the RHS. This implies that the silent validator never checks in the equilibrium. The same holds even if we add the active validator's deposit to the reward of a silent validator. The indifference condition in this case becomes:

$$\pi(1-\alpha)(R+L) = C. \tag{16}$$

However, the mechanics change when we consider more than 1 active validator. Suppose there are m silent validators. The indifference condition of such a validator is different from the active validator, as it does not risk losing stake if it does not check. On the other hand, if the silent validator checks while no staked validator does, it will be rewarded both by staked validator stakes – nL – and the dishonest Asserter's stake R. Silent validators have the same cost of checking, C. By a similar argument as with only active validators, it is easy to show that there is no pure Nash equilibrium solution to the game. The proof is by contradiction: if one of the validator types checked with certainty, it would make malicious Asserter not make a false claim, causing validators not to check. Therefore, we again consider a totally mixed Nash equilibrium solution. The probability that the silent validator plays the checking strategy is β. Then, the indifference condition for the silent validator is:

$$C = \pi(1-(1-\alpha)^n)\sum_{i=0}^{m-1}\left(\binom{m-1}{i}\beta^i(1-\beta^{m-1-i})\frac{R+nL}{i+1}\right). \tag{17}$$

The indifference condition for active validator is the same as (10), as for this type of validator it does not matter what silent validators will do. If there is a

[2] The role of a silent validator can be played by the Asserter as well. That is, instead of stealing all the assets in the system, it may only collect nL and allow the system to survive.

false claim and active validators do not find it, they will lose all their stakes. For completeness, we state the condition here:

$$\pi\left(\sum_{i=0}^{n-1}\binom{n-1}{i}\alpha^i(1-\alpha)^{n-1-i}\frac{R}{i+1}\right)-C=\pi(P_{s,\alpha}(n-1)(-s_w)+(1-P_{s,\alpha}(n-1))(-L)). \tag{18}$$

The indifference condition for the malicious Asserter is:

$$(1-(1-\alpha)^n)(1-(1-\beta)^m)R=((1-\alpha)^n+(1-\beta)^m-(1-\alpha)^n(1-\beta)^m)U. \tag{19}$$

Analyzing indifference conditions (17), (18) and (19) gives conditions on the parameters when totally mixed equilibrium of the game exists. Consider, for example, $m=2$ and $n=1$. The indifference conditions become:

$$C=\pi\alpha(R+2L), C=\pi(R+L), (2\alpha-\alpha^2)\beta R=(1-(2\alpha-\alpha^2)\beta)U.$$

This solves $\alpha=\frac{R+L}{R+2L}$, $\beta=\frac{U}{(2\alpha-\alpha^2)(R+U)}$ and $\pi=\frac{C}{R+L}$. That is, α needs to be high enough, to make sure that β is smaller than 1. When α is low enough, then it must be that $\beta=1$. That is, silent validators always check.

3 Protocol Level Incentives

In this section, we ask the question of how to reward validators for checking (and posting about) the true claim.[3] The post, a transaction to a smart contract at the base layer network does not need to include proof. This approach allows obtaining a pure strategy equilibrium, in which all validators check with certainty. This guarantees that a false claim is found with probability one. However, it comes at the cost of adding new functionality to the protocol, which is usually not desirable.

Similar to the previous section, there are $n+1$ validators. The probability that each one needs to post about checking the state of the world is denoted by P. Validators need to take a decision whether to check or not before they find out whether they need to post about the result. In case they fail to post when they are required to post, they have slashed their stakes L. The cost of checking is C, as before. The cost of posting is c, which is typically assumed to be less than C. The opportunity cost of staking on the platform is rL in each round. Therefore, r denotes the return on investment outside the system in one round. The payment validators receive for posting the right outcome is denoted by p. Then, the expected payoff is equal to $-C+P(-c+p)$ when the validator checks, or $P(-L)$ when the validator does not check. To guarantee that the validator checks in equilibrium, the expected payoff of checking needs to be larger than the expected payoff of not checking:

$$-C+P(-c+p)>P(-L).$$

[3] For a similar discussion for Ethereum validator incentivization see https://dankradfeist.de/ethereum/2021/09/30/proofs-of-custody.html.

This gives a condition on P, namely $P > \frac{C}{p-c+L} =: \pi_l$.

The expected budget of the protocol per round is equal to nPp, which is lower bounded by $n\pi_l p = \frac{nCp}{p-c+L}$. Note that taking a high enough L lowers the expected cost of the system to guarantee *incentive compatibility (IC)*, but it increases the cost to guarantee *individual rationality (IR)*. The latter means that validators want to be a part of the system in the first place, instead of staying away and obtaining zero utility. To guarantee IR, we need to offset the opportunity cost the validator incurs by staking L, which equals rL. Since by IC, the validator always checks, we need that $-C + P(-c + p) > rL$, that is, $P > \frac{C+rL}{p-c} =: \pi_r$. This simplifies to the condition that π needs to be larger than $\max(\pi_c, \pi_r)$. The minimum value of π is achieved at the minimax. One function is decreasing in L, another is increasing. The minimax is achieved when they are equal. That is,

$$\frac{C + rL}{p - c} = \frac{C}{p - c + L},$$

implying $L = \frac{rc-rp-C}{r}$ (which is negative, therefore, not possible) or $L = 0$. This optimization is done for fixed p. We can minimize P and Pp over p as well. Plugging in $L = 0$ into the formula of P gives: $P = \frac{C}{p-c}$. It is minimized for p as large as possible. Similarly, $pP = \frac{Cp}{p-c}$ is minimized for p as large as possible. The value approaches C, which is intuitive:

1. the system pays exactly the cost of checking on average,
2. it checks the validators with very low probability,
3. when they are checked—the system pays a very large amount p.

Unless there is some cost associated with high payment p for the protocol to upper bound it, this is an optimal strategy. However, such costs are obvious. The protocol cannot invest an arbitrarily high amount at once in rewarding validators.

Implementation

We present the implementation of attaching a message of checking and posting with some probability to an assertion. The sampling can be done on the protocol level, by referring to state-relevant hash values. Suppose the Asserter is making a claim about the value of $f(x)$ for some function f which is common knowledge, and a value x which varies across different runs of the protocol. We want to pose a randomly generated challenge to the validator, such that the checker must know $f(x)$ in order to respond correctly to the challenge. Then we can punish the validator for responding incorrectly.

The validator has a private key k, with a corresponding public key g^k which is common knowledge (g is a suitable generator of a group where the Diffie-Hellman problem is hard). To issue a challenge for the computation of $f(x)$, the asserter generates a random value r, then publishes (x, g^r) as a challenge. A validator who has private key k should respond to the challenge by posting a tiny transaction on-chain if and only if $H(g^{rk}, f(x)) < T$, where H is a suitable hash function and T is a suitably chosen threshold value.

Note that only the Asserter (who knows r) and the validator (who knows k) will be able to compute the hash because they are the only two parties who can compute g^{rk}. Note also that computing the hash requires knowledge of $f(x)$. After the validator(s) have had a window of time to post their response(s) to the challenge, the Asserter can post its claimed $f(x)$ which will be subject to challenge if any validator disagrees with it.

At this time, the Asserter can accuse any validator who responded incorrectly: the Asserter must publish r to substantiate its accusation. If the Asserter's claimed value of $f(x)$ is later confirmed, a smart contract can verify the accusation and punish the misbehaving validator (if the Asserter's claimed $f(x)$ is rejected, the Asserter's accusation is ignored).

If any funds are seized from validators, the Asserter gets half of the seized funds and the remainder is burned. One way to build this in the rollup is to have assertions, rather than revealing the state root $f(x)$. Instead, include an attention challenge $(x, g^r, H(x, g^r, f(x)))$, which is also a binding commitment to $f(x)$, and only reveal $f(x)$ when there is a challenge, or when the assertion is confirmed. Validators could self-identify and stake, and they would have until the confirmation time of the assertion to post their response to the attention challenge.

4 Conclusions and Future Work

We initiate a study of the optimal number of validators and their stake sizes in the rollup protocols. The main result is that for higher system security guarantees, the cost of checking should be low, TVL should be high and the number of validators should be as low as possible. We also derive optimal validation and assertion deposits in the equilibrium. Future avenues of research include weighted staking. Even if such staking is not allowed, if one validator creates multiple identities, but checks only once, it results in weighted staking. Such validator's indifference condition is different from the others, as it has invested kL tokens, where k is the number of copies it created.

References

1. Asharov, G., Canetti, R., Hazay, C.: Towards a game theoretic view of secure computation. In: Kenneth G. Paterson, editor, Advances in Cryptology - EUROCRYPT 2011–30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15–19, 2011. Proceedings, volume 6632 of Lecture Notes in Computer Science, pp. 426–445. Springer (2011)
2. Brünjes, L., Kiayias, A., Koutsoupias, E., Stouka, A.-P.: Reward sharing schemes for stake pools. In: IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7–11, 2020, pp. 256–275. IEEE (2020)
3. Gersbach, H., Mamageishvili, A., Pitsuwan, F.: Decentralized attack search and the design of bug bounty schemes. CoRR, arXiv:abs/2304.00077 (2023)
4. Gersbach, H., Mamageishvili, A., Schneider, M.: Staking pools on blockchains. CoRR, arXiv:abs/2203.05838 (2022)

5. Kalodner, H.A., Goldfeder, S., Chen, X., Matthew Weinberg, S., Felten, E.W.: Arbitrum: scalable, private smart contracts. In: Enck, W., Felt, A.P., (eds.), 27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15–17, 2018, pp. 1353–1370. USENIX Association (2018)
6. Li, J.: On the security of optimistic blockchain mechanisms. Available at SSRN 4499357 (2023)
7. Luu, L., Teutsch, J., Kulkarni, R., Saxena, P.: Demystifying incentives in the consensus computer. In: Ray, I., Li, N., Kruegel, C., (eds.), Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12–16, 2015, pp. 706–719. ACM (2015)
8. Tas, E.N., Boneh, D.: Cryptoeconomic security for data availability committees. In: Forthcoming at Financial Cryptography (2023)



Characterizing Common Quarterly Behaviors in DeFi Lending Protocols

Aaron Green(✉), Michael Giannattasio, Keran Wang, John S. Erickson, Oshani Seneviratne, and Kristin P. Bennett

The Rensselaer Institute for Data Exploration and Applications, Rensselaer Polytechnic Institute, Troy, USA
{greena12,giannm,wangk16,erickj4,senevo,bennek}@rpi.edu

Abstract. The emerging decentralized financial ecosystem (DeFi) is comprised of numerous protocols, one type being lending protocols. People make transactions in lending protocols, each of which is attributed to a specific blockchain address which could represent an externally-owned account (EOA) or a smart contract. Using Aave, one of the largest lending protocols, we summarize the transactions made by each address in each quarter from January 1, 2021, through December 31, 2022. We cluster these quarterly summaries to identify and name common patterns of quarterly behavior in Aave. We then use these clusters to glean insights into the dominant behaviors in Aave. We show that there are three kinds of keepers, i.e., a specific type of users tasked with the protocol's governance, but only one kind of keeper finds consistent success in making profits from liquidations. We identify the largest-scale accounts in Aave and the highest-risk kinds of behavior on the platform. Additionally, we use the temporal aspect of the clusters to track how common behaviors change through time and how usage has shifted in the wake of major events that impacted the crypto market, and we show that there seem to be problems with user retention in Aave as many of the addresses that perform transactions do not remain in the market for long.

1 Introduction

Decentralized Finance (DeFi) is an emerging economic ecosystem built on blockchain technologies and smart contracts. The primary feature differentiating DeFi from traditional finance is the lack of intermediaries controlling financial services. DeFi's growth in the last half-decade has started with the creation of protocols that seek to replicate the services of traditional financial institutions. For example, a popular kind of DeFi protocol is the lending protocol, which mimics the functionality of a bank from traditional finance.

Lending protocols in DeFi give users the opportunity to lend their crypto assets to a lending pool, effectively acting as a savings account. Users can then borrow crypto assets from the lending pool based on how much they have

http://idea.rpi.edu.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
P. Pardalos et al. (Eds.): MARBLE 2023, 2024.
https://doi.org/10.1007/978-3-031-48731-6_4

contributed to the pool themselves. Thus, the lending pool is inherently over-collateralized. Borrowed assets will accrue interest on the loan, and a portion of that interest is paid back to the lenders of the assets. In this way, lenders can accrue "deposit" interest on the money they put into the protocol. If a borrower's collateral loses too much value or the loan accrues too much interest, so their account no longer meets the minimum collateral requirements set by the protocol, their account is subject to being liquidated. In this case, a third party, called a "keeper," can pay off some of the loans to acquire a portion of the borrower's collateral. The amount of collateral purchased through a liquidation often comes with a small "liquidation bonus" that acts as an incentive for keepers to perform these liquidations and keep the lending pool healthy.

Having decentralized protocols that allow for these actions creates new ways for retail and institutional users to try leveraging their assets for profit. Whether someone wants to accrue passive interest on their crypto assets or whether they want to use the collateralized borrowing enabled by lending protocols to pursue more aggressive, riskier positions, lending protocols represent an important part of the emerging DeFi economy. One of the major lending protocols is Aave.[1] [8,9,13] As of February 20, 2023, Aave has over $6.87 billion of assets locked in its lending pools across two versions and seven markets. At its peak in October 2021, Aave had over $18 billion of crypto assets locked in its lending pool.[2]

Using the transaction data of Aave's V2 [9] deployment on the Ethereum blockchain [10] as the primary subject of this work, we summarize quarterly user and smart-contract transactions on the platform. We then cluster these summaries to identify the predominant types of behaviors in Aave. We use these clusters to present novel insights about the behavior of keepers, the trends of the largest accounts (likely owned by financial institutions), and the highest risk behaviors on Aave. We also track how the clusters change over time and in response to some of the major events that had affected the crypto market, such as when China announced an impending crypto ban in May 2021 [1] and the November 2022 crash of major crypto exchange FTX [2]. Finally, we examine related work in the field and discuss the potential impacts of this analysis and how this can be used to help future work.

2 Methods

2.1 Data Sources

We examine DeFi behavior at the address level since addresses are the equivalent of accounts in traditional finance. Every DeFi transaction is associated with an address which may be either an externally-owned account (EOA) or a smart-contract. To identify and understand common quarterly user behavioral patterns, we cluster quarterly representations of addresses associated with their transactions. Thus we need data that summarizes the behavior that has

[1] www.aave.com.
[2] see https://defillama.com/protocol/aave-v2.

been associated with each address in a quarter. This is different from the raw data that we collect, which is transaction-level data. We briefly describe the transaction-level data, and then explain how we convert this data into address-level summaries. To distinguish between EOA's and smart-contacts, we obtained blockchain address data from Amberdata[3] for each address present in our data to classify them as either an EOA or a smart contract.

Code used to collect these data can be found on a public GitHub repository.[4] The transaction-level data from TheGraph is freely accessible, but any data acquired from Amberdata requires one's own API key. The code used to query Amberdata is included, but with no API key.

2.2 Transaction-Level Data

The transaction-level data we use in this analysis comes from The Graph.[5] We use data from the lending protocol Aave, which consists of six transaction types: deposits, collaterals, withdraws, borrows, repays, and liquidations. Liquidation transactions involve two addresses: one associated with the keeper and the other with the liquidatee. Because performing liquidations and being liquidated are both interesting, we duplicate the liquidation transactions and for the two copies, treat the liquidator as the subject in one and the liquidatee as the subject in the other. This effectively turns the liquidation transactions into "liquidations performed" transactions and "liquidated" transactions. Each transaction has data regarding the time the transaction occurred, the address that initiated the transaction, the asset(s) involved in the transaction, the amount of the asset(s) used in the transaction (in both native amounts and USD amounts), and any relevant third-party addresses like the liquidatee. We use all transactions that occurred between January 1, 2021 and December 31, 2022, for a total of two years of transaction data. This amounts to 1,665,737 total transactions involving 172,872 unique blockchain addresses.

2.3 Address-Level Summaries

From this transaction data, we group the transactions by the address that initiated the transaction and the quarter during which the transaction took place. For each groups of address-quarters, we create the summary features in Table 1.

2.4 Computation of Clusters

To cluster this data, we use a fuzzy c-means algorithm [7]. We use the R Programming language [22], and compute the clusters using the `ppclust` package

[3] www.amberdata.io.
[4] https://github.com/aaronmicahgreen/Characterizing-Common-Quarterly-Behaviors-In-DeFi-Lending-Protocols.
[5] www.thegraph.com.

Table 1. Address-quarter summary features derived from transaction-level data used for clustering

Feature Name	Description
Smart Contract?	Binary value classifying the address as an EOA (0) or a smart contract (1)
Number of Withdrawals	Number of withdraw transactions this account performed during the quarter
Value of Withdrawals	Value (in USD) of assets this address withdrew from its account during the quarter
Number of Repayments	Number of repay transactions this address performed during the quarter
Repaid Value	Value (in USD) of the assets this address repaid during the quarter
Percentage of Stable Borrows	Percentage of this address' borrows that used the stable borrow rate during the quarter
Number of Borrows	Number of borrow transactions this account performed during the quarter
Borrowed Value	Value (in USD) of the assets this address borrowed during the quarter
Mean Transaction Value	Mean amount (in USD) of each transaction made by this address during the quarter
Number of Deposits	Number of deposit transactions this account performed during the quarter
Deposited Value	Value (in USD) of assets this address deposited into its account during the quarter
Number of Collateral Transactions	Number of collateral transactions this account performed during the quarter
Number of Days Active	Number of days in the quarter during which the address posted at least one transaction
Number of Transactions	Number of transactions performed by this address during the quarter
Liquidations Performed	Number of liquidation transactions that this address performed in the quarter
Liquidations Performed Value	Value (in USD) of assets that this address liquidated during the quarter
Times Liquidated	Number of transactions that liquidated this address during the quarter
Liquidated Value	Value (in USD) of assets that were owned by this address and that were liquidated during the quarter
New User?	Binary value marking whether this address is new this quarter (1) or not (0)
Number of Active Collateral Assets	Number of unique assets posted as collateral by this address at the end of the quarter

[11]. To select the number of clusters, we combined the results from two methods: the elbow method [27] and the Silhouette scores [24]. The elbow method indicated that 5 or 8 clusters would work well, and the silhouette score for 2 and 8 clusters were the highest. For these reasons, we chose to use 8 clusters. Since the transaction data tends to be heavy tailed, we scaled the data using the `lambertW0` function from the "lamW" package [5] prior to clustering.

3 Results

The discovered clusters of quarterly behavior show quite distinct traits. A heatmap of the features within the clusters can be seen in Fig. 1. This row-scaled heatmap was created in R using the `pheatmap` package [17]. In this heatmap, darker red features indicate that the feature has higher values within that cluster compared with other clusters, and vice versa with darker blue features. For clarity, the "Smart Contract?" feature will appear darker red when the given cluster is composed of a higher proportion of smart contract addresses, and the "newUser?" feature will appear darker red when the cluster is composed of a higher proportion of addresses that are new in the quarter.

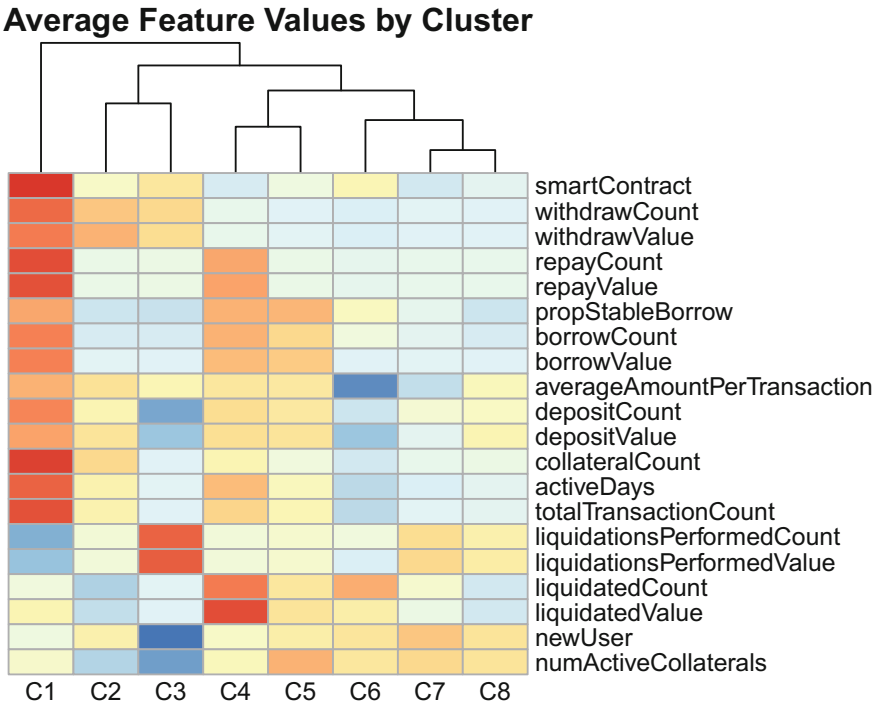


Fig. 1. Heatmap showing the relative values of features across clusters of behavior associated with individual addresses within quarters

3.1 Interpretations of Clusters

Inspecting the properties of each cluster, we provide names and interpretations of each cluster in Table 2. The number of address-quarter behaviors that fit into each cluster is also given, which shows a fairly balanced spread of behavior across Aave's history. We give justification of these names and interpretations here, making use of Figs. 1, 3, and 2, as well as some numerical data when we feel that raw numbers help clarify a point.

We use the term "retail" to describe behavior that we believe is performed by individual users or their smart contracts on a smaller scale. We use this term in a similar way to the term "retail investor" in traditional finance, which is used to describe non-professional investors trading through brokerage accounts or their own savings. This is in contrast to "institution" accounts, which we hypothesize to be addresses that are run by professional organizations such as banks or hedge-funds and are transacting with significantly higher amounts than retail addresses.

Table 2. Names, counts, and descriptions of clusters of quarterly behaviors in Aave from January 1, 2021—December 31, 2022

Cluster Number	Count	Name	Description
C1	15,291	Whales	High activity, high-value contracts that emphasize creating and re-balancing complex but safe positions
C2	16,842	Retail Savers	Medium volume and value deposits and withdraws within a single quarter
C3	18,435	Experienced Keepers	Knowledgeable, high-value users and their contracts that find profitable liquidation opportunities
C4	14,060	Highest-Risk Behavior	EOAs with high amounts of borrows and repays and whose accounts were liquidated the most
C5	14,024	Yield Farming	A mix of contracts and EOAs depositing and borrowing mid-value positions
C6	34,082	Inactives	Low-to-no-activity EOAs closing their lingering positions through repayments or being liquidated
C7	33,803	New Keepers	New accounts with low activity overall but higher-than-average liquidations performed
C8	20,093	Retail Keepers	EOAs who try to find available liquidations for small profits

We chose the name "Whales" for C1 because this cluster has the highest proportion of smart contracts (19.38%), and the average amount per transaction in this cluster is the highest ($1.19 million per transaction, compared to the overall average of $564,129). To pair with huge transactions, the addresses in this cluster also make the most transactions, making an average of 33.12 transactions per quarter, which is more than double the average of the next highest cluster.

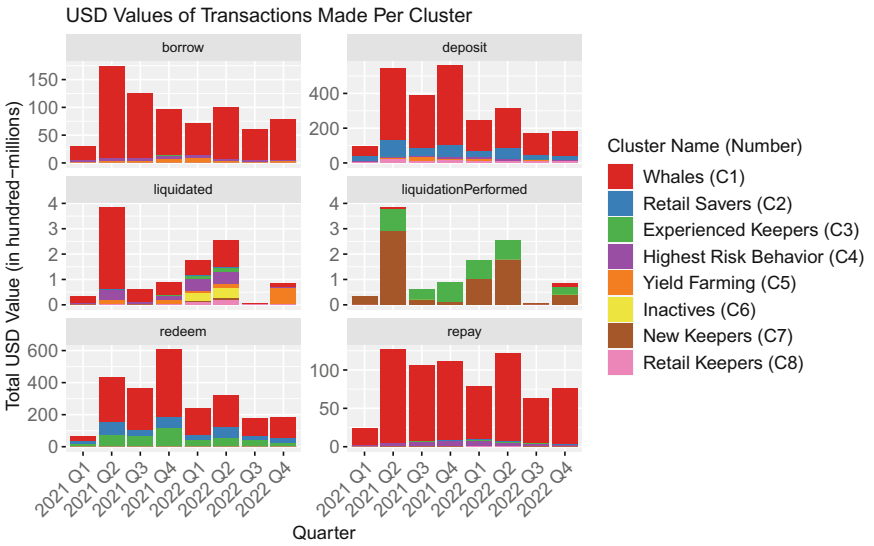


Fig. 2. Set of bar plots showing the USD value of the transactions each cluster made in each quarter, with a separate bar plot for each transaction type

We choose the name "Retail Savers" for C2. Figure 1 shows this cluster is dominated by its deposits and withdrawals. These accounts perform few-to-no borrows (and likewise, few-to-no repays), and also have the lowest likelihood of any cluster for being liquidated. These factors all indicate that this cluster contains behaviors reminiscent of retail accounts that are either seeking to deposit their crypto-assets into the lending pool to accrue deposit interest, or withdraw their assets to exit the lending pool.

We chose the name "Experienced Keepers" for C3. This cluster accounts for the vast majority of the liquidations performed and the high-value liquidations, and the C3 addresses have done little else. This cluster has the second-highest proportion of smart contracts (13.05%), and 83.77% of the transactions performed by this cluster were by smart contracts. which was expected for keepers because performing liquidations needs to be done through contracts (see [8]). This cluster contains very few new addresses, so most of these keepers have been active prior to the quarter in which they fall into this cluster.

We choose the name "Highest Risk Behavior" for C4 because this cluster's most notable characteristic is the number of times its accounts are liquidated, as well as the value of those liquidations. Addresses in this cluster are far more likely to be liquidated than in any other cluster. They also tend to borrow high amounts without depositing very much, which is exactly the kind of behavior that is expected to lead to liquidations.

We chose the name "Yield Farming" for C5 because this cluster has high proportions of borrows and deposits, which is reminiscent of how yield farmers leverage deposited assets to repeatedly borrow and deposit assets in order to

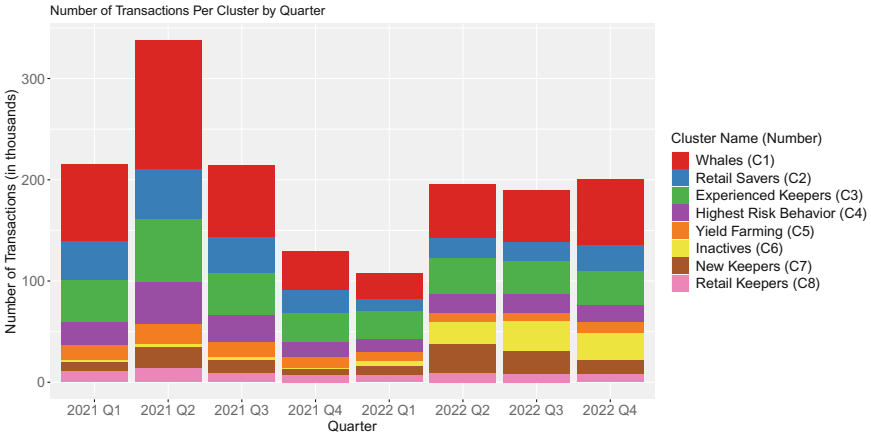


Fig. 3. Bar plot showing the amounts of transactions that addresses in each cluster performed in each quarter from 2021 Q1 through 2022 Q4

accrue higher amounts of interest on their deposits. Leveraging assets in this way also increases the riskiness of these accounts' positions, making them more likely to be liquidated in the event of higher-than-expected price volatility with the currencies they have used. Predictability is key when creating high-risk positions, so borrowing with stable borrow rates allows for better control over their positions, and indeed this cluster has a high proportion of stable-interest-rate borrows.

We choose the name "Inactives" for C6 because the number of transactions that these addresses perform in a quarter is the lowest across all clusters. Addresses in this cluster only perform an average of 2.67 transactions in the quarter. Additionally, these addresses are making deposits the least of any transactions, and their accounts are liquidated with relatively high frequency. This cluster contains an above-average number of new users, too. It likely represents both addresses of users who just did a couple of small transactions to test out the functionality of the platform and also accounts that are passively accruing interest. With the higher number of times being liquidated, some of these accounts are likely holding onto positions that have become unhealthy while the account owners either did not care to rebalance their positions, have been priced out of rebalancing due to high transaction fees [20], or have just done poor jobs of monitoring their account health. Despite this cluster being characterized by low transaction counts, we see from Table 2 that this is the largest cluster over the course of the eight quarters. Because there are so many addresses that fall into this cluster, the number of transactions performed by this cluster is non-negligible, as seen in Fig. 3, but these transactions are of such low value that they do not make any impact on the combined transaction values in Fig. 2.

We choose the name "New Keepers" for C7 because this cluster's most notable features are the number of liquidations they perform and the value

of those liquidations, which leads to the "keeper" label. These addresses also have the highest tendency to be new in the quarter when they are assigned this cluster, leading to the label "new."

Finally, we choose the name "Retail Keepers" for C8 because the most notable features of this cluster are the number of deposits, the value of deposits and the liquidations performed. This cluster is very similar to C7, but the amount and value of the liquidations performed by this cluster are lower, and the number and value of deposited assets are higher. We hypothesize that these accounts are retail accounts that are trying to make some deposit interest and who are occasionally trying to perform some liquidations, but at low success rates. They do not tend to be smart contracts (only 5.18% are smart contracts), which indicates that these are likely users who are searching for potential accounts to liquidate manually instead of in an automated way, and perhaps cannot afford to liquidate larger accounts in their entirety since they do not tend to have high-value accounts.

3.2 Insights Derived From Clusters

Now that we have an understanding of what the clusters mean, we show how these clusters can be used to gain meaningful insights into the DeFi market. We focus on three main insights from these clusters. First, we discuss the "Whales," which are the addresses that account for the largest portion of money spent in Aave. Then we discuss the different clusters of keepers and how they have shifted over time. We conclude with a discussion of the overall trend in Aave usage and how the platform seems to have poor user retention overall.

3.2.1 Risk-Averse Whales: Cluster 1 contains the addresses that are often known as "Whales." This is the cluster that performs the most transactions, and also makes transactions of the highest amounts. We hypothesize that this cluster consists predominantly of accounts owned by various financial institutions. It seems likely that, should an institution such as a bank or hedge fund decide to try building a profitable position in DeFi, they would have a tendency to do so through smart contracts they have written. Additionally, we would expect such an institution to have much more capital than individual users. We might also expect these institutions to be more risk-averse, and this cluster accounts for a low proportion of the total number of times liquidated despite having the largest open positions in the market.

Mueller discusses in [20] the effects of transaction fees on markets in DeFi, and how periods of time with higher transaction fees can affect some users' abilities to react to changing market conditions. Since institutions have access to more capital and are holding onto larger market positions, they are not going to be priced out of re-balancing or closing positions when the market shifts. Additionally, these institutions likely have the most knowledge of how to safely act in financial markets, and have better access to financial data that will allow them to preemptively change their positions to avoid getting liquidated. We see

these features represented in this cluster, as the addresses in this cluster are consistently making the highest number of transactions and accounting for the bulk of the USD amount of all transactions in Aave, showing their propensity to be undeterred by transaction fees. However, because these accounts have such large positions, the few liquidations that these accounts experience account for a large portion of the total amount liquidated.

3.2.2 Keepers' Behaviors in Aave: One of the most interesting facets of this clustering is that it gives some key insights about keepers in Aave. With over $69 million in total profits made through liquidations in these two years, it makes sense that there would be many people seeing this opportunity and trying to insert themselves into the pool of keepers who turn big profits. This is why the second-most-populous cluster is the "new keepers" cluster, and the "retail keepers" cluster is the third largest. However, these clusters have a high propensity of only being active for a single quarter. 75.94% of "new keepers" and 55.19% of "retail keepers" are only active for a single quarter. The successful accounts among these two clusters do often end up in the "experienced keepers" cluster in later quarters, with over 15% of, both "new keepers" and "retail keepers" eventually becoming "experienced keepers".

Of these three clusters, the "new keepers" and "experienced keepers" account for the vast majority of the amount of USD that is actually liquidated. The 33,803 accounts in the "new keepers" cluster liquidated accounts worth a combined total of $676 million, and the 18,435 accounts in liquidated accounts worth a total of $387 million total. These numbers are in stark contrast to the "retail keepers" cluster, which does contain accounts that perform liquidations more frequently than most other clusters, but which only total about $2,782 worth of liquidations performed. Despite the large amount of total money that has been available through liquidations in this two-year period in Aave, retail users have not tended to make much money out of liquidations, especially if they are reliant on manually monitoring Aave accounts to find potential liquidations. Combined with the fact that the three clusters we classify as containing keepers contain over 43.4% of all addresses across these two years, this indicates that many DeFi users are more interested in trying to profit off of the risky behavior of other users rather than use DeFi platforms for their other functionality. Since the profits of keepers are not distributed very evenly across those trying to perform liquidations, this is likely a contributing factor to the high amount of user churn we discuss more in the next section.

3.2.3 Cluster Changes Over Time and Issues With User Retention: Through this eight-quarter stretch, the number of addresses in each cluster of behavior per quarter can be visualized with the Sankey plot in Fig. 4. Two additional "clusters" have been added to this visualization to help show the flow of address behavior from one quarter to the next: the "future active addresses" cluster (shown in orange) and the "addresses with no new activity" cluster (shown in pink). "Future active addresses" is a strictly-decreasing cluster that contains

the addresses that were observed making transactions in Aave during any of the eight quarters, but that has not done so yet. Since every address in our dataset makes at least one transaction, this cluster contains no addresses in 2022 Q4. "Addresses with no new activity" contains addresses that have made transactions in a past quarter, but made no transactions in the quarter in question. Understandably, there are no addresses present in this cluster in 2021 Q1.

Each of the 8 computed clusters shows fairly consistent sizes throughout these eight quarters, with the exception of 2022 Q4. It is interesting to see through this two-year span of Aave how many of the addresses fall into the "addresses with no new activity" group, which would seem to indicate a problem with user retention on the platform. This user-retention problem stems from at least a couple of factors. First, as discussed earlier when examining the various clusters of keepers, there are many accounts that surface in Aave which seem to solely be attempting to break into the liquidation market. The market appears fairly saturated, though, as many of the new keepers do not turn significant profits and end up only acting in a single quarter. The largest liquidation spikes in Aave's history came from two events: the China announcement of an impending ban on cryptocurrencies in May 2021 and the failure of the Terra Luna blockchain in May 2022. We see in Fig. 2 that the quarters when these events took place (2021 Q2 and 2022 Q2, resp.) correspond to the largest amounts of liquidations. Looking at Fig. 4, we also see increases in "new keepers" and "retail keepers" cluster sizes in both of these quarters, indicating that the events which cause high amounts of liquidations bring with them an influx of new accounts trying to profit from the liquidations, but that most do so unsuccessfully. Since these accounts make up over 40% of accounts that have transacted within Aave over this time span, these are certainly one driving factor for apparent user-retention problems.

Interestingly, despite the liquidation spike events, they have not caused any significant changes in the quarterly number of addresses that have transacted on Aave. However, there was a significant decline in the number of active addresses in the final quarter of 2022. Every cluster fell in size except for "retail savers" (C2) in 2022 Q2, and the largest cluster is also the least active (C6). This decline in the final quarter of our data is likely due to the sudden collapse of what was one of the largest crypto exchanges, FTX, in November 2022. It will be interesting to continue this analysis in 2023 to see whether the usage of Aave has recovered following this event.

It is also worth noting the general trend of the cryptocurrency market over these two years. For much of 2021, the crypto market was increasing. It reached an all-time high in early November 2021, which is near when Aave had its highest value locked of over $18 billion. Since then, however, the crypto market as a whole has dipped significantly. By July of 2022, the total market was down about 75% from its peak, and for the remainder of the year, it remained steady around that same level. Considering the usage of Aave did not drop proportionally with the value in the crypto market, this could indicate that there is a market for users

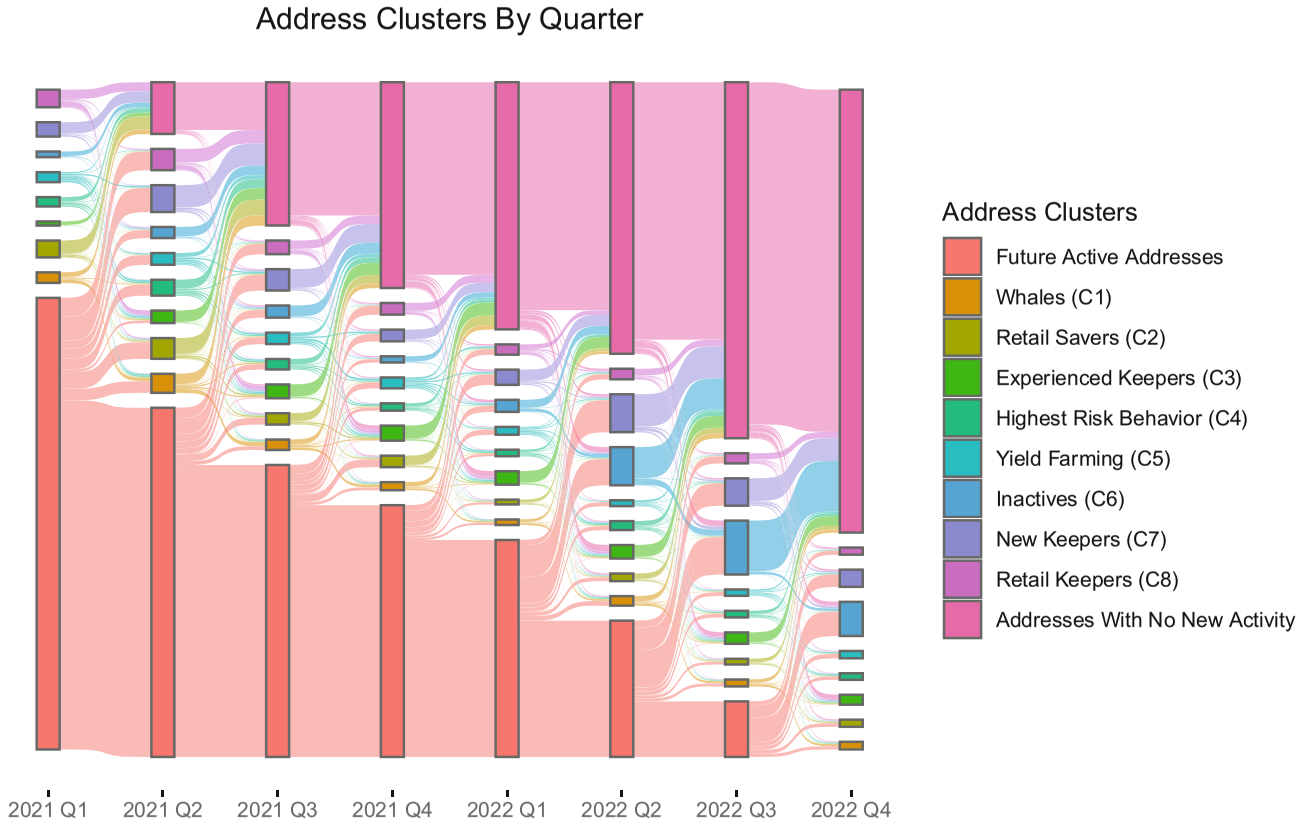


Fig. 4. Sankey diagram showing how addresses changed clusters from quarter to quarter colored by cluster. Orange represents address with activity in later quarters. Pink represents addresses with no activity in that quarter

who see value in crypto and DeFi beyond simply trying to make a profit, which would be a very positive note.

4 Related Work

There has been other work seeking to characterize user behavior in lending protocols and DeFi. For instance, in Green et al. [14] a process is created for converting the transaction data into a form suitable for the application of survival-analysis methods. This allows for the macro-level analysis of micro-level events, showing how different covariates like whether a borrowed coin is a stable coin affect the time it takes for users to repay the borrowed coin. Such a focus on transaction sequences and the time between events could prove interesting in conjunction with the clustering analysis presented in this paper.

Some work has been done on a small scale to compute other features of account-level data, such as their end-of-day market positions and their end-of-day overall health factor [20]. DeFi lending requires over-collateralized loans, so at any point, a borrower should have collateral in their account that exceeds their debt. The ratio of their collateral to their debt makes up the "health factor" of their account, and if the health factor drops below a certain threshold, this is when the debt position is available to be liquidated. Knowing an account's health factor would be useful for characterizing the risk that an account is willing to take on. For instance, maybe one account will re-balance its positions to consistently keep its health factor near 1.5, whereas a second account aims to keep its health factor near 2.0. In this case, we could characterize the first account as riskier than the second, because they intentionally operate with a lower health factor. Computing or acquiring the health factors of accounts through time could be a very useful feature for more informative clustering. Similarly, Qin et al. [21] have analyzed risk management provided by keepers that act on accounts within lending protocols. They have measured various risks that liquidation participants are exposed to on four major Ethereum lending pools (i.e., MakerDAO [19], Aave [8], Compound [23], and dYdX [6]), notably including how borrowers ought to monitor their loan-to-value ratios in order to make timely changes to their account positions in order to try and avoid being liquidated.

Another facet of lending that could be useful for further understanding behavioral patterns in DeFi is the account usage of stablecoins. Stablecoins behave much differently in the crypto market than non-stable coins, as they have nearly constant value (typically they are pegged to the US dollar, and so coins like USDC, USDT, and DAI have held steady right around $1 for years). This property of stablecoins can be exploited in lending protocols to help create positions whose health factors are more predictable. For example, if an account takes out a loan using stablecoins as collateral, then the health of the account should only vary with the relative price of the principal asset as opposed to a more complicated relationship between the prices of the principal and collateral assets. Kozhan and Viswanath-Natraj [18] provide some early empirical evidence on the effects of stablecoin-backed loans in DeFi, and have found relationships between

the loan risk and price volatility of the DAI. Quantifying how accounts use stablecoins in their borrowing and lending patterns could be another interesting feature (or set of features) to help more accurately characterize behaviors in DeFi lending.

5 Discussion and Future Work

The crypto market and the associated DeFi ecosystem have been extraordinarily volatile for as long as it has existed. Our clustering of address-level behavior on a quarterly basis helps better understand how usage has changed in the wake of major events that shake up the markets, such as when China announced a ban on cryptocurrencies in May 2021[1], when the Terra Luna blockchain crash in May 2022 [3], and the FTX fraud discovery and subsequent collapse in November 2022 [2].

Every time one of these events occurs, we see sizable shocks in the market. Besides watching prices drop, it is interesting to see how people have actually reacted to shock events in their patterns of usage. The May 2021 shock from China's announcement of an impending crypto ban caused the largest spike in liquidations, which can be seen in Fig. 2 2021 Q2, and there has been a mostly steady decline in the amount that has been borrowed quarter-over-quarter since then. Similarly, the next largest spike in liquidations occurred in May 2022 after the Terra Luna crash. However, the most visible change in usage patterns seems to have occurred in 2022 Q4, which is when FTX crashed. This crash seems to have many DeFi enthusiasts and institutions just waiting to see what happens next in the market. Due to the huge financial losses suffered by investors through the FTX crash (estimated at over $8 billion [4] and the likelihood that this event leads to new regulations for DeFi, it seems many former DeFi users are more pensive regarding how or whether to engage in DeFi for the time being. Correlating these shock events to changes in observed behavior proves interesting as it can validate our own intuitions and also reveal surprising trends that deviate from our expectations.

In addition to external shock events, many DeFi protocols are built with internal governance mechanisms that allow their own user base to propose and make changes to the operations of the protocol as a group. In Aave, these proposals tend to involve setting protocol-level values for things like individual cryptocurrency loan-to-value ratios, liquidation thresholds, or which coins are allowed to be used as collateral (see [8,9,13]). These governance changes are also enacted in the form of blockchain transactions, and currently, some of them are available on Amberdata. Incorporating these transactions into our existing stream and seeing how behavior within protocols changes following governance changes would also be interesting. Whether such changes in a protocol lead to significant, or even noticeable changes to behavior could help DeFi developers build more effective governance mechanisms in new protocols.

Our work in this paper focused solely on the Aave V2 Ethereum market, but this is just one of many Aave markets. Aave has been deployed on Avalanche

[25], Polygon [15], Optimism [28], Fantom [12], Harmony [26], and Arbitrum [16], and on some of these it has been deployed with multiple versions. One next step for this work is to see how well the clusters examined in this paper hold up across the other Aave markets. Each market of Aave likely appeals to different groups for one reason or another. For instance, Polygon has significantly lower transaction fees than the Ethereum market, and thus we would expect to see higher transaction volumes among retail users who would be penalized less for making higher-frequency transactions. Should the clustering hold up well in other Aave markets, it would then be interesting to see how well they apply to the other large lending protocols like Compound [23] and MakerDAO [19]. Expanding the scope of data into decentralized exchanges (DEXes) would also be useful. DEXes account for a large portion of transactions in DeFi, and platforms like Uniswap and Sushiswap are some of the most popular in the DeFi ecosystem. Classifying common behavioral patterns in DEX usage would help in seeing the bigger picture of overall DeFi usage, and it may even be possible to find addresses that are present in more than one platform and cluster their behavior across multiple platforms.

The code for computing the clusters used in this paper, as well as for creating the figures, can be found in a public github repository.[6][7]

Acknowledgments. The authors acknowledge the support from NSF IUCRC CRAFT center research grants (CRAFT Grants #22003, #22006) for this research. The opinions expressed in this publication and its accompanying code base do not necessarily represent the views of NSF IUCRC CRAFT. This work was supported by the Rensselaer Institute for Data Exploration and Applications (IDEA). We also would like to thank Amberdata for providing some of the data used in this work.

References

1. www.fortune.com/2022/01/04/crypto-banned-china-other-countries/
2. www.investopedia.com/what-went-wrong-with-ftx-6828447
3. www.coindesk.com/learn/the-fall-of-terra-a-timeline-of-the-meteoric-rise-and-crash-of-ust-and-luna/
4. www.theguardian.com/business/2023/jan/11/ftx-fraud-value-crypto-sbf
5. Adler, A.: lamW: Lambert-W Function (2015). https://doi.org/10.5281/zenodo.5874874,www.CRAN.R-project.org/package=lamW, r package version 2.1.1
6. Antonio Juliano: dYdX: A Standard for Decentralized Margin Trading and Derivatives. Tech. rep. (09 2017), www.whitepaper.dydx.exchange/
7. Bezdek, J.C., Ehrlich, R., Full, W.: Fcm: The fuzzy c-means clustering algorithm. Comput. Geosci. **10**(2), 191–203 (1984). https://doi.org/10.1016/0098-3004(84)90020-7, www.sciencedirect.com/science/article/pii/0098300484900207
8. Boado, E.: AAVE Protocol Whitepaper. Technical report (01 2020). www.cryptocompare.com/media/38553941/aave_protocol_whitepaper_v1_0.pdf

[6] https://github.com/aaronmicahgreen/Characterizing-Common-Quarterly-Behaviors-In-DeFi-Lending-Protocols.

[7] https://idea.rpi.edu/.

9. Boado, E.: AAVE Protocol Whitepaper V2.0. Technical report (12 2020). www.cryptorating.eu/whitepapers/Aave/aave-v2-whitepaper.pdf
10. Buterin, V.: Ethereum: A next-generation smart contract and decentralized application platform (2014). www.github.com/ethereum/wiki/wiki/White-Paper
11. Cebeci, Z.: Comparison of internal validity indices for fuzzy clustering. J. Agricult. Inf. (2), 1–14 (2019). https://doi.org/10.17700/jai.2019.10.2.537
12. Choi, S.M., Park, J., Nguyen, Q., Cronje, A.: Fantom: a scalable framework for asynchronous distributed systems (2018). https://doi.org/10.48550/ARXIV.1810.10360, www.arxiv.org/abs/1810.10360
13. Frangella, E., Herskind, L.: AAVE V3 Technical Paper. Technical report (01 2022). www.github.com/aave/aave-v3-core/blob/master/techpaper/Aave_V3_Technical_Paper.pdf
14. Green, A., Cammilleri, C., Erickson, J.S., Oshani, Seneviratne, Bennett, K.P.: Defi survival analysis: insights into risks and user behaviors (2022). www.marble-conference.org/marble2022-cfp
15. Kanani, J., Sandeep Nailwal, A.A.: Matic network whitepaper. www.github.com/maticnetwork/whitepaper (2020)
16. Kalodner, H.A., Goldfeder, S., Chen, X., Weinberg, S.M., Felten, E.W.: Arbitrum: Scalable, private smart contracts. In: USENIX Security Symposium (2018)
17. Kolde, R.: pheatmap: Pretty Heatmaps (2019). www.CRAN.R-project.org/package=pheatmap, r package version 1.0.12
18. Kozhan, R., Viswanath-Natraj, G.: Decentralized stablecoins and collateral risk. WBS Finance Group Research Paper Forthcoming (2021)
19. MakerDAO: The Maker Protocol: MakerDAO's Multi-Collateral Dai (MCD) System. Technical report. www.makerdao.com/en/whitepaper/#abstract
20. Mueller, P.: Defi leveraged trading: Inequitable costs of decentralization (2022). www.dx.doi.org/10.2139/ssrn.4241356
21. Qin, K., Zhou, L., Gamito, P., Jovanovic, P., Gervais, A.: An empirical study of defi liquidations: Incentives, risks, and instabilities. In: Proceedings of the 21st ACM Internet Measurement Conference, pp. 336–350 (2021)
22. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2022). www.R-project.org/
23. Leshner, R., Hayes, G.: Compound: The money market protocol. Technical report. (02 2019). www.compound.finance/documents/Compound.Whitepaper.pdf
24. Rousseeuw, P.J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**, 53–65 (1987). https://doi.org/10.1016/0377-0427(87)90125-7, www.sciencedirect.com/science/article/pii/0377042787901257
25. Sekniqi, K., Laine, D., Buttolph, S., Sirer, E.G.: Avalanche platform (2020). www.assets.website-files.com/5d80307810123f5ffbb34d6e/6008d7bbf8b10d1eb01e7e16_Avalanche%20Platform%20Whitepaper.pdf
26. Team, H.: Harmony technical whitepaper (2020). www.harmony.one/whitepaper.pdf
27. Thorndike, R.: Who belongs in the family? Psychometrika **18**(4), 267–276 (1953). https://doi.org/10.1007/BF02289263
28. Tyneway, M.: Optimism (2020). www.github.com/ethereum-optimism/optimism



Blockchain Transaction Censorship: (In)secure and (In)efficient?

Zhipeng Wang(✉), Xihan Xiong, and William J. Knottenbelt

Imperial College London, London, England
zhipeng.wang20@imperial.ac.uk

Abstract. The ecosystem around blockchain and Decentralized Finance (DeFi) is seeing more and more interest from centralized regulators. For instance, recently, the US government placed sanctions on the largest DeFi mixer, Tornado.Cash (TC). To our knowledge, this is the first time that centralized regulators sanction a decentralized and open-source blockchain application. It has led various blockchain participants, e.g., miners/validators and DeFi platforms, to censor TC-related transactions. The blockchain community has extensively discussed that censoring transactions could affect users' privacy. In this work, we analyze the efficiency and possible security implications of censorship on the different steps during the life cycle of a blockchain transaction, i.e., generation, propagation, and validation. We reveal that fine-grained censorship will reduce the security of block validators and centralized transaction propagation services, and can potentially cause Denial of Service (DoS) attacks. We also find that DeFi platforms adopt centralized third-party services to censor user addresses at the frontend level, which blockchain users could easily bypass. Moreover, we present a tainting attack whereby an adversary can prevent users from interacting normally with DeFi platforms by sending TC-related transactions.

Keywords: Security · Blockchain · Censorship · Mixer · DoS attacks

1 Introduction

On August 8th, 2022, the US Treasury's Office of Foreign Assets Control (OFAC) placed sanctions [17,18] on the largest zero-knowledge proof (ZKP) mixer, Tornado.Cash (TC) [1], due to alleged facilitation of money laundering. TC has been used to process more than 7B USD worth of cryptocurrencies since its creation in 2019. OFAC added the TC website and related blockchain addresses to the "Specially Designated Nationals And Blocked Persons" (SDN) list. According to the sanctions, US citizens are no longer legally allowed to use the TC website or involve any property or interest transactions with those blacklisted addresses. To our knowledge, this is the first time that centralized regulators sanction a decentralized and open-source blockchain application.

The sanctions have led to a series of consequences. For instance, the largest prior-merge Ethereum mining pool, Ethermine, stopped processing any TC

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
P. Pardalos et al. (Eds.): MARBLE 2023, 2024.
https://doi.org/10.1007/978-3-031-48731-6_5

deposit and withdrawal transactions since August 9th, 2022 [3]. Many DeFi platforms, e.g., Uniswap [19], Aave [2], and dYdX [5], have started banning addresses that receive transactions from TC after the sanctions were announced. Centralized transaction propagation services, e.g., Front-running as a Service (FaaS) such as Flashbots, also ban transactions calling OFAC-blacklisted addresses. Circle, the issuer of the stablecoin `USDC`, has already frozen all `USDC` held in OFAC-blacklisted TC addresses [3].

However, the sanctions of an open-source DeFi application operating on top of blockchains, bring up ew questions and extensive discussions in the blockchain community [6] and even within the US government [16]. Privacy advocates argue that the banning of ZKP mixers violates citizens' right to privacy, and OFAC exceeds its statutory authority by treating an autonomous and decentralized application as an individual or entity.

In this paper, we study blockchain censorship from a novel perspective. We investigate if it is possible to achieve "fine-grained" censorship on permissionless blockchains to fully ban tainted transactions. We analyze the censorship during the life cycle of blockchain transactions (i.e., generation, propagation, and validation) to reveal the efficiency and security implications of censoring transactions and addresses. We summarize our contributions as follows:

1. **Censorship Reduces Miners' Security:** We investigate how blockchain miners censor transactions. Our results indicate that users can easily bypass miners' current censorship. Therefore, we propose a fine-grained censoring algorithm. However, we prove that censorship will reduce miners' security, because an adversary can craft tainted transactions to attack miners. We show that the attack comes at zero gas fees when all miners adopt censoring.
2. **Dissect FaaS Censorship Mechanism:** We analyze the blockchain transaction censorship during the propagation process in FaaS. By analysing the relayed blocks by six FaaS (i.e., Flashbots, Eden Network, Manifold, Aestus, Agnostic Gnosis and Blocknative), we find that only Flashbots is complying with OFAC regulations by censoring TC-related transactions. However, our analysis indicates that fine-grained censorship will also reduce FaaS's security.
3. **Bypassing DeFi Platform Censorship:** We analyze how DeFi platforms (e.g., Uniswap, dYdX, and Aave) ban user addresses. We find that DeFi platforms leverage centralized third-party services to censor user addresses at the frontend level. Therefore, users can resort to using intermediary addresses or a command line interface (CLI) to bypass the censorship. Additionally, we present an attack whereby an adversary can deliberately taint innocent addresses by sending transactions involving blacklisted addresses. This attack can prevent users from interacting normally with DeFi platforms' frontend.

2 Background

2.1 Blockchain and Smart Contracts

Blockchains [12,23] are distributed ledgers on top of a global peer-to-peer (P2P) network. Users can join and leave the network freely. There is no central authority

to guarantee common agreement on the distributed ledgers among users. Users can achieve agreement through Sybil-resilience mechanisms, such as Proof-of-Work (PoW) for prior-merge Ethereum, and Proof-of-Stake (PoS) for post-merge Ethereum. Smart contracts are quasi-Turing-complete programs which can be executed within a virtual machine. Users can leverage smart contracts to build DeFi services [4,21]. Transactions are propagated over a public P2P or a private relay network. Miners and non-mining traders can manipulate the transaction order and front-run other traders by unilaterally determining the order or paying higher transaction fees to extract Blockchain Extractable Value (BEV) [14].

2.2 Centralized Transaction Propagation Services

Independent of the P2P network, emerging centralized relay services, i.e., FaaS, offer an alternative option for users to bid for the priority to extract BEV by communicating to miners/validators privately. For example, on Flashbots, traders can add an arbitrary number of signed transactions (including transactions from other parties) to their bundle, along with metadata specifying the bundle execution logic. Traders can then submit transaction bundles directly to miners/validators without a broadcast on the P2P network. Auctions through Flashbots are risk-free, meaning unsuccessful bids do not need to pay transaction fees.

2.3 ZKP Mixers

ZKP mixers, inspired by Zerocash [15], are one of the most widely-used privacy solutions for non-privacy-preserving blockchains. ZKP mixers are running on top of smart-contract-enabled blockchains, e.g., Ethereum. Upon using a mixer, a user deposits a fixed denomination of coins into a pool and later withdraws these coins to another address [1,9,20]. When used properly, ZKP mixers can break the linkability between addresses, thus enhancing users' privacy. Therefore, ZKP mixers, e.g., TC [1], are widely used for money laundering [20] and receiving the initial funds to launch on-chain attacks [24].

2.4 Blockchain Regulation and Censorship

Although permissionless blockchains, such as Bitcoin and Ethereum, seem to be able to evade regulation and censorship through their decentralization, their surrounding ecosystem has attracted interest from regulators [8,11]. Regulators have started enforcing existing financial regulations for off-chain services, such as anti-money laundering (AML) regulations for centralized exchanges. Though off-chain regulations will not directly affect on-chain activities, blockchain participants may follow regulations to ban transactions related to specific addresses. For instance, on August 8th, 2022, the US OFAC announced sanctions against TC, and added TC-related addresses to the SDN List [17,18]. To the best of our knowledge, this is the first time that centralized regulators sanction a decentralized application. After the announcement, some Ethereum miners, FaaS, and DeFi platforms have started censoring TC-related transactions and addresses [3].

3 System Model

In this section, we outline our system and threat model for blockchain censoring.

3.1 System Components

Address: On permissionless blockchains, a user has at least one public/private key-pair, which corresponds to their *address* and controls cryptocurrencies.
Smart Contract: Smart contracts are quasi-Turing-complete programs that typically execute within a virtual machine. A smart contract function can be called by a transaction, and can also call the functions of other contracts. A smart contract can emit *events* when successfully being executed.
Transaction: To transfer assets or trigger the execution of smart contract functions, a user signs a transaction with its private key. The transaction's sender pays for the cost of the triggered smart contract execution, i.e., transaction/gas fees. An *internal transaction* is a transaction triggered by a smart contract as a result of one or more previous transactions.
Block: A block includes a list of transactions. A blockchain consists of a growing list of blocks, which are securely linked together using cryptographic hashes.
Miners/Validators: Miners on PoW blockchains, or validators on PoS blockchains, are responsible for: *(i)* sequencing transactions, i.e., specifying the order of transactions within a block; *(ii)* verifying transactions and blocks; *(iii)* confirming transactions and proposing blocks; and *(iv)* propagating data. In this paper, we regard the terms "miner" and "validator" as interchangeable.
Blockchain Network: Blockchains are operating on top of a global P2P network. Users can join, exit, and discover other nodes in the network. A transaction can be propagated over the network. Moreover, users can leverage centralized transaction propagation services, i.e., FaaS, to transmit a transaction directly to miners, without broadcasting it to the remaining network.

3.2 Blockchain Censoring

We introduce the fundamental components of blockchain censoring as follows.
Blacklisted Addresses: Blacklisted addresses are a list of blockchain addresses (including smart contract addresses) that are banned by off-chain regulators (e.g., the US OFAC). Users are legally prohibited from involving any property or interest transactions with those addresses.
Tainted Transactions: We define a transaction as tainted if it *(i)* is issued by a blacklisted address, *(ii)* transfers assets to a blacklisted address, or *(iii)* triggers a function of a blacklisted smart contract address.
Transaction Censorship: We define *transaction censorship* as an action to prevent a tainted transaction from being generated, propagated, or validated.
Censorship Categories. Figure 1 shows the life cycle of a transaction. A transaction can be censored at different steps (i.e., generation, propagation, and validation). In the following, we list the various censorship categories.

- *Generation censoring*: A censoring blockchain application (e.g., Centralized Exchange or DeFi platform) will ban the interaction between their frontend and the addresses which *(i)* attempt to generate tainted transactions, or *(ii)* interacted with blacklisted addresses.
- *Propagation censoring*: A censoring FaaS or a P2P node will not choose to forward any tainted transactions to miners/validators or other P2P nodes.

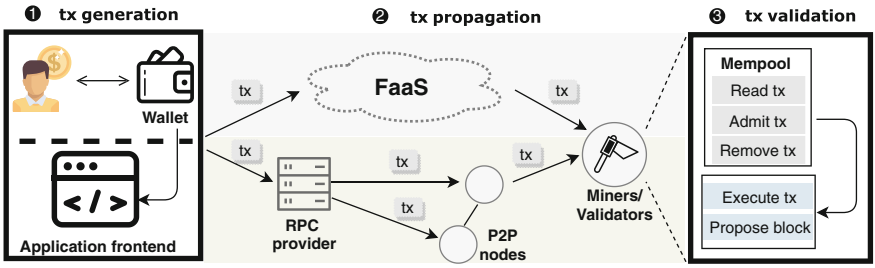


Fig. 1. Blockchain transaction life cycle. In step ❶, a user's wallet creates a transaction tx, which involves generating a signature with the user's private key. This generation can be performed locally or interact with a blockchain application's frontend. In step ❷, the wallet sends tx to a RPC provider, which will broadcast tx into the entire P2P network. The wallets can also send tx to a FaaS, which will forward tx to validators via a private network. In step ❸, upon receiving tx, a validator will first collect the transaction into its mempool, and will then include tx into a newly proposed block after verifying tx. A transaction can be censored in the steps of generation, propagation, or validation.

- *Validation censoring*: A censoring miner/validator will not include tainted transitions in their proposed blocks. However, a censoring miner/validator can receive blocks which contain tainted transactions and are proposed by others.

3.3 Threat Model

Given a censoring blockchain participant, which can be a miner/validator, FaaS, or application, the adversary's goal is to *(i)* bypass the participant's censorship on tainted transactions, or *(ii)* attack the participant to prevent it from executing or forwarding non-tainted transactions.

We further assume that the adversary possesses the following capabilities:
Sending Private Transactions: The adversary can send an unconfirmed transaction directly to miners or FaaS via their private RPC, or propagate the transaction over the remaining P2P network.
Crafting Complicated Transactions: The adversary can create a tainted transaction in which multiple contracts are called. The time for executing the transaction increases over the number of contracts.

4 Censorship During Transaction Validation

In the following, we investigate miners' censorship, and propose a DoS attack against the censoring miners through crafting sophisticated transactions.

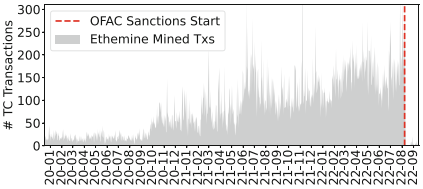


Fig. 2. TC deposit and withdrawal transactions mined by Ethermine over time. Ethermine stopped processing TC transactions during August 10th and August 24th, 2022.

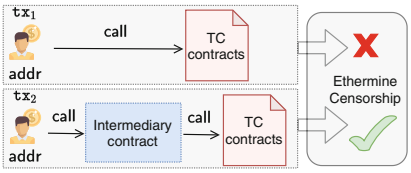


Fig. 3. Ethermine bans the transaction that calls TC contracts directly (e.g., tx_1). However, if a user leverages an intermediary contract to interact with TC, then the transaction (e.g., tx_2) will pass Ethermine's censorship.

4.1 Miners' Censorship on Tainted Transactions

The OFAC sanctions against TC have had an influence on Ethereum miners. As shown in Fig. 2, we plot the distribution of TC transactions mined by the largest mining pool, Ethermine, before the Ethereum merge, i.e., September 15th, 2022. We observe that, from August 10th to August 24th, 2022, Ethermine stopped processing any transactions related to deposits and withdrawals in TC (cf. Fig. 2). This indicates that Ethermine censors TC-related transactions. Interestingly, we find that 1 deposit and 98 withdrawal transactions can still bypass Ethermine's censorship after August 24th, 2022.

Miners' Censorship on TC. To understand how Ethermine censors TC-related transactions, we perform the following experiments and analysis.

- We create a transaction tx_1 that calls TC contracts directly, and send tx_1 to Ethermine through its private RPC.[1] We then observe that tx_1 will not be mined.
- We analyze the 99 transactions which bypassed Ethermine's censorship after August 24th, 2022. We find that all these transactions first call an intermediary contract, which is not a blacklisted address, and the intermediary contract will later call blacklisted TC contracts through internal transactions.

As shown in Fig. 3, we can thus infer that, Ethermine's censorship on blacklisted addresses works as follows: *(i)* Upon receiving a pending transaction tx, Ethermine checks tx's from-address and to-address. *(ii)* If the 3from-address or

[1] https://ethermine.org/private-rpc, available on September 1st, 2022.

Algorithm 1: Fine-grained censorship on tainted transactions.

Input: tx: an unconfirmed transaction
Output: True or False: tx will be mined or not
Param: $\{\texttt{addr}\}_{ban}$: set of blacklisted addresses
1 **if** *tx's* from-address $\in \{addr\}_{ban}$ *or tx's* to-address $\in \{addr\}_{ban}$ **then**
2 | **return** False
3
4 Execute tx locally, and record the set of called addresses, $\{\texttt{addr}\}_{call}$.
5 **if** $\{\texttt{addr}\}_{call} \cap \{\texttt{addr}\}_{ban} \neq \emptyset$, **then return** True; **else return** False

to-address of tx is an OFAC-blacklisted address, i.e., a blacklisted address is directly called in tx, then Ethermine evicts tx from its mempool for pending transactions.

Improving Miners' Censorship Mechanism. Figure 3 depicts that Ethermine does not censor the transactions that call TC contracts through internal transactions, which means users can still interact with TC using intermediary contracts. Generally, blacklisted addresses can be called through internal transactions, and a censoring miner has to execute a transaction to check if it is tainted. We thus propose the following claim (cf. Claim 1).

Claim 1. *Given an unconfirmed transaction tx, to check if it is tainted, a censoring node can simply simulate tx locally.*

Based on Claim 1, we propose a novel algorithm to check tainted transactions. As shown in line 1–3 of Algorithm 1, we keep Ethermine's style of censorship as the first step to filter the transactions from a blacklisted address or calling blacklisted addresses directly. Moreover, given a transaction tx, to check if any blacklisted addresses are called through internal transactions in tx, miners execute tx locally to extract all called addresses and check whether they are blacklisted (cf. line 1–4 in Algorithm 1). Therefore, Algorithm 1 can censor all tainted transactions.

4.2 DoS Censoring Miners Through Crafting Tainted Transactions

In the following, we investigate the downside of the improved censorship algorithm (cf. Algorithm 1), which can enable an adversary to attack censoring miners.

Censoring Computation Cost. We craft a TC-related transaction with multiple intermediary contracts (cf. Fig. 4), and the TC contract is called by the last contract. For each intermediary contract, we add time-consuming operations. We then evaluate the crafted transactions' execution time when calling

a various number of intermediary contracts. We adopt Hardhat to deploy the contracts and execute the transactions locally on an Ethereum Erigon node. The node is running on a macOS Ventura machine with an Apple M1 chip with 8-core CPU, 8-core GPU, 16-core Neural Engine, 16 GB of RAM, and 2 TB SSD storage in configuration. Figure 5 shows that the execution time approximately increases linearly over the number of intermediary contracts.

DoS Censoring Miners. The expensive censoring computation cost brings up the opportunities for the adversary to attack miners. Intuitively, the adversary can craft numerous complicated and tainted transactions, and keep sending them to the victim miner. Therefore, the victim will be exhausted from censoring those complex transactions and cannot process new non-tainted transactions.

Attack Strategy. We propose the following strategy to DoS censoring miners: *(i)* The adversary crafts m complicated and tainted transactions. Each of those transactions is configured with a high gas price, which is *much higher* than any existing transactions in the victim's mempool. *(2)* The adversary then keeps sending the m crafted transactions to the victim, via the victim's private RPC. We provide the attack results in a private Ethereum network in the full version of this paper.[2]

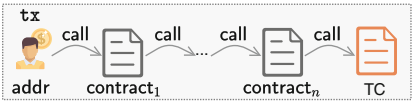


Fig. 4. In a crafted TC transaction, n intermediary contracts are called consecutively, and the final intermediary contract will call TC contracts.

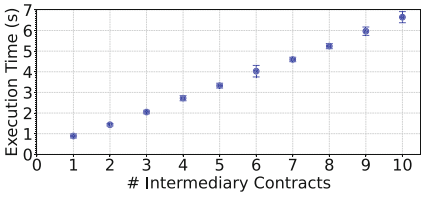


Fig. 5. A crafted transaction's execution time on a local node when calling multiple intermediary contracts.

4.3 Attack Cost

We analyze the attack costs when DoSing censoring miners in different cases.

Case 1: All Miners Adopt Censoring. If all miners adopt censoring, then none of the tainted transactions will be mined. Therefore, the adversary does not need to pay any transaction fees. The attack cost will only be the cost of buying attack machines, electricity, network bandwidth consumption, etc. We assume that these costs are constant and denote their sum as $\mathcal{C}_{cnst}$.

Attacking All Miners Simultaneously. As shown in Fig. 6, instead of attacking a specific miner, the adversary can DoS all censoring miners simultaneously. The adversary can broadcast a crafted and tainted transaction tx to the entire

[2] https://eprint.iacr.org/2023/786.pdf.

P2P network (rather than send tx via a specific miner's private RPC), and every miner will finally receive tx. In this case, all censoring miners will waste their computation power on checking tx but will not mine it. If the adversary broadcasts sufficient tainted transactions with a high gas price, then all censoring miners could be DoSed. Note that the attack cost comes at zero transaction fees because no tainted transactions will be mined.

Case 2: Non-zero But Not All Miners Adopt Censoring. In this case, some miners do not choose to adopt censoring, and a tainted transaction might be successfully mined. The adversary might suffer from the cost of gas fees.

As shown in Fig. 7, when a censoring miner receives a transaction tx and finds that tx is tainted, then the miner can forward tx to its peers. Finally, a non-censoring miner will receive tx and mine it. Therefore, the adversary has to pay the transaction fee of tx.

Attacking A Single Censoring Miner. Assume that the average fee for a crafted transaction tx is f. If the censoring miner forwards tx to its non-censoring peers, then the adversary needs to pay $f \cdot m + \mathcal{C}_{cnst}$. Recall that m is the number of transactions that the adversary sends to the miner. However, if the censoring miner just abandons the crafted transactions, then the attack cost is $\mathcal{C}_{cnst}$.

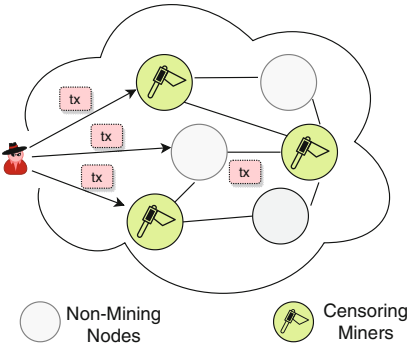


Fig. 6. Attacking all miners simultaneously when all miners adopt censorship. No tainted transaction will be mined, and the adversary does not need to pay any transaction fees.

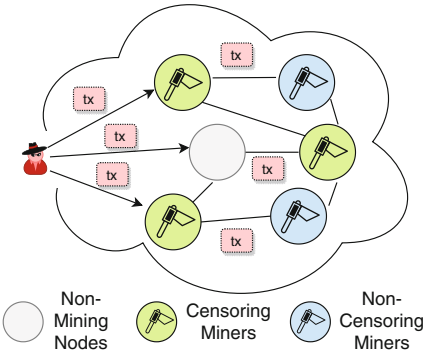


Fig. 7. Attacking multiple censoring miners simultaneously when there are non-censoring miners. The adversary will pay a cost if tainted transactions are mined by non-censoring miners.

Attacking Multiple Censoring Miners Simultaneously. The adversary can also attempt to broadcast the tainted and complicated transactions to the entire P2P network. Analogously, the cost of transaction fees is determined by the number m of crafted transactions and the average transaction fee f, i.e., $f \cdot m$. Therefore, the total attack cost is $cost = \mathcal{C}_{cnst} + f \cdot m$. Moreover, a crafted transaction tx might be first mined by a non-censoring miner before any censoring miners receive it. Although censoring miners will not pre-execute tx to check if it is tainted, they still need to spend computation source on validating the block including tx.

5 Censorship During Transaction Propagation

This section analyzes FaaS's censorship during transaction propagation.

5.1 FaaS Workflow

We take Flashbots [22] as an example to analyze the workflow of FaaS. Figure 8 describes the transaction order flows before the Ethereum merge. ❷ denotes the public user order flow where a user sends its transactions to a public node (e.g., Infura) RPC. In contrast, ❶ shows the private user order flow, where a user switches to a private RPC endpoint to protect its transaction from being front-/back-run by adversaries [4]. ❸ depicts the searcher order flow, where a searcher listens to public transactions propagated over the P2P network. Once finding BEV opportunities, the searcher constructs a bundle of transactions in an immutable order and submits it to the Flashbots relay.

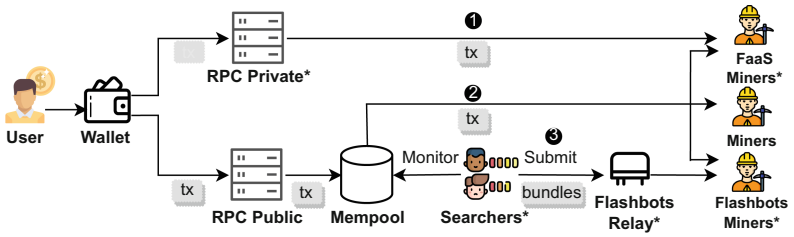


Fig. 8. Flashbots transaction order flows pre-PBS. ❶, ❷ and ❸ denote private, public and search order flows. * denotes entity with potential censoring power.

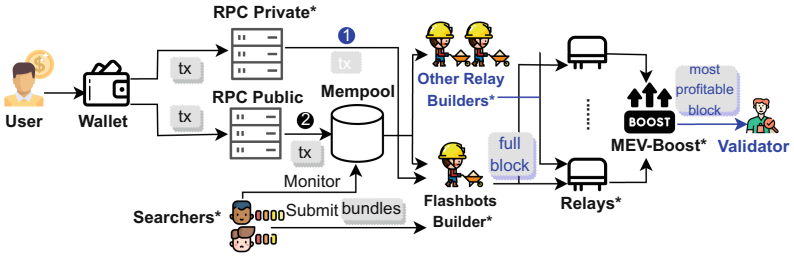


Fig. 9. Flashbots transaction order flows post-PBS. ❶, ❷ and ❸ denote private, public and search order flows. * denotes entity with potential censoring power.

However, several important changes happened after the Ethereum merge. Specifically, Flashbots implements a protocol named Proposer–Builder Separation (PBS) via *MEV-Boost*, which separates the block construction role from the

block proposal role. PBS allows validators (i.e, *proposers*) to outsource the block-building roles to specialized parties called *builders*. As shown in Fig. 9, searchers submit bundles to builders, which are responsible for building full blocks with available transactions and submitting bids to relays. A relay verifies the validity of the execution payload and selects the most-profitable block sent by all connected builders, while the MEV-Boost picks the best block from multiple relays. The block proposer receives blind blocks, signs the most profitable block, and sends it back to the relay. Once verifying the proposer's signature, the relay responds with the full block for the proposer to propose to the network. Note that although MEV-Boost mitigates the centralization of the validator set, it may cause the builder centralization, e.g., the dominant builder with the highest inclusion rate may receive exclusive order flows from users and searchers [7].

5.2 FaaS Censorship Mechanism

To understand how FaaS adopts censorship, we conduct empirical analysis by crawling all blocks relayed by six FaaS through their public APIs, i.e., Flashbots, Eden Network, Manifold, Aestus, Agnostic Gnosis and Blocknative from the Ethereum block 15,537,940 (September 15th, 2022) to 16,331,031 (January 4th, 2023). We also crawl the transactions in which TC deposit or withdrawal events are emitted. As shown in Figures 10 and 11, we plot the total number of relayed

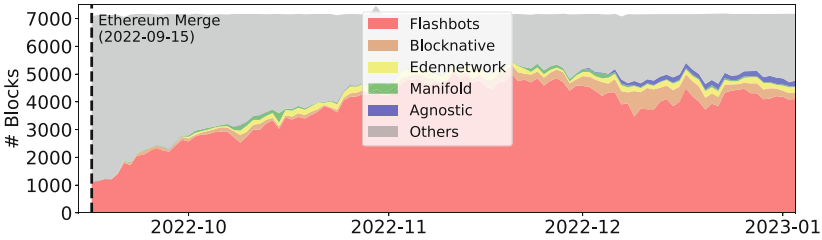


Fig. 10. Distribution blocks relayed by different FaaS after the Ethereum Merge. Flashbots relay the most blocks, i.e., more than 52.54% of the total blocks during September 15th, 2022 and January 4th, 2023 are relayed by Flashbots.

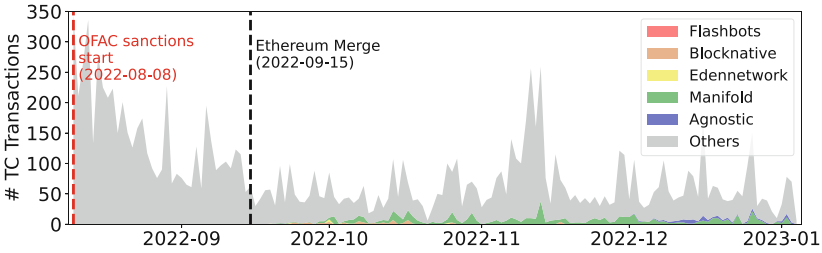


Fig. 11. Distribution of TC deposit and withdrawal transactions from August 8th, 2022 to January 4th, 2023. Flashbots do not relay any blocks including TC-related transactions after the Ethereum block 15537940 (September 15th, 2022).

blocks, and the relayed blocks which include TC-related transactions. We observe that, although Flashbots relay the most blocks during the timeframe (cf. Fig. 10), none of TC-related transactions are relayed by Flashbots (cf. Fig. 11). This result indicates that Flashbots ban TC deposit and withdrawal transactions.

We take Flashbots as an example to dissect its censorship. After checking the code in its GitHub repositories, we find that Flashbots complied with OFAC regulations by censoring TC-related transactions in the following ways:

Period 1: From OFAC sanction announcement to the merge. First, the Flashbots RPC endpoint censors the user private order flow (i.e., ❶ in Fig. 8). It configures the blacklisted TC-related addresses, and checks whether the to-address or from-address in the transaction are blacklisted. The transaction will be censored if any blacklisted address is found. In addition, the Flashbots Relay censors the searcher order flow (i.e., ❸ in Fig. 8) by checking the blacklisted addresses in the received bundles. It is worth noting that a searcher or miner in Fig. 8 may also censor transactions by simulating the transaction execution.

Period 2: Post merge. Additionally, Flashbots has a Block Validation Geth client to help censor TC-related transactions. In contrast to the Flashbots RPC endpoint and the Flashbots Relay, which simply check whether the TC contracts are called directly, the Block Validation Geth client checks all the intermediary contract calls in a given transaction execution trace, i.e., adopting the fine-grained censorship in Algorithm 1. Similarly, a builder or searcher in Fig. 9 still has potential censoring power. In contrast, a validator can not censor transactions since it receives a blind block from the MEV-Boost.

5.3 DoS Censoring FaaS Searchers and Builders

In the following, we discuss the potential DoS attacks against censoring searchers and private RPCs in the post-merge FaaS system based on the following assumptions: *(i)* the adversary is the user in Fig. 9; *(ii)* searchers and builder are honest because they have to maintain their reputations; and *(iii)* the censoring entities perform fine-grained check (cf. Algorithm 1) on blacklisted addresses.

Attack Strategy. Similar to attacking a censoring miner in Sect. 4.2, the adversary can create numerous tainted transactions by adding a call to a blacklisted address after calling several intermediary contracts consecutively (cf. Fig. 4). Based on the assumption *(iii)*, an adversary controlling multiple addresses can launch a targeted attack against a given private RPC, or even a non-targeted attack against searchers. Note that we do not discuss the possibility of attacking relays, as we assume that searchers and builders are honest.

6 Censorship During Transaction Generation

In this section, we analyze DeFi platform censorship during the transaction generation process. For completeness, we refer the reader to the full version of this paper for other blockchain components' censorship.

6.1 Non-transparent Frontend-Level Censorship

Although DeFi protocols are running on a decentralized blockchain, the websites interacting with the protocols are centralized. The entities that develop and maintain the websites will risk breaking the law if they do not follow the OFAC sanctions. However, the centralized entities' censorship is non-transparent (cf. Fig. 12). In the following, we leverage public information to analyze the platforms' censorship, which claims to follow the OFAC sanctions.

Table 1. TC user addresses interacting with DeFi platforms from September 15th, 2022 to January 4th, 2023.

Platform	Censorship start date	TC user addresses	
		Depositors	Withdrawers
Uniswap	Before 2022/08/23 [19]	88	213
Aave	2022/08/10 [2]	2	5
dYdX	2022/08/10 [5]	1	0

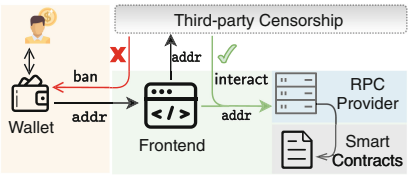


Fig. 12. DeFi platforms generally leverage a third-party service to determine whether an address should be banned.

Uniswap. Uniswap is a Decentralized Exchange running on top of Ethereum. Uniswap claims that they cooperate with TRM Labs to identify on-chain financial crime and block addresses that are owned or associated with clearly illegal behaviors such as sanctions, terrorism financing, hacked or stolen funds, etc [19]. Therefore, the OFAC-sanctioned TC addresses are also banned by Uniswap Decentralized Application (DApp). Although it was reported that Uniswap has prohibited 253 addresses on its frontend, so far as we understand, Uniswap and TRM Labs do not intend to publish their censoring mechanism and data.
Aave. Aave is an on-chain lending platform. Similar to Uniswap, Aave leverages TRM Labs to determine financial crime and other prohibited activities [2]. Although Aave provides an API for users to check whether their addresses will be banned, Aave does not disclose the censorship details on their IPFS frontend.
dYdX. dYdX is a DeFi platform supporting perpetual, margin and spot trading, as well as lending and borrowing. dYdX has confirmed that it blocked several addresses in line with the OFAC's sanctions against TC [5]. dYdX claims they have long utilized "compliance vendors" to identify sanction-related addresses. However, dYdX's censorship is still not public at the time of writing.

6.2 Investigating DeFi Platforms' Censorship

To investigate how a DeFi platform censors blacklisted addresses, we leverage on-chain data to analyze if TC user addresses can still interact with the DeFi platform after the OFAC sanctions are announced.
Post-Sanction TC User Addresses. We crawl the addresses that are used to deposit and withdraw in the four TC `ETH` pools after the OFAC sanctions

are announced. We identify 2,282 TC user addresses during September 15th, 2022 and January 4th, 2023, out of which 805 are used to deposit and 1,581 to withdraw. For these 2,282 addresses, we crawl their historical transfers of `ETH` and ERC20 tokens. We also crawl 379 labeled addresses from Etherscan of the three censoring DeFi platforms. We finally analyze whether the 2,282 TC user addresses interact with the platforms after they deposit/withdraw into/from TC.
Results. As shown in Table 1, we identify that 89 deposit and 216 withdrawal addresses can still interact with the three censoring platforms. For instance, on October 29th, 2022, the address 0x2d7...7F7 withdrew from TC 100 `ETH` pool at block 15,853,770 and then swapped 54.5 `ETH` to 88,509 `USDC` on Uniswap at block 15,853,783. These results indicate that the existing DeFi platforms' censorship mechanism is inefficient and may cause false negatives.

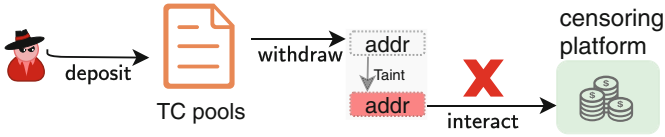


Fig. 13. Tainting attack overview. After depositing into TC, the adversary assigns an innocent address as the withdrawal address. Then the address will be tainted, and thus blocked when interacting with the censoring DeFi platforms' frontend.

6.3 Tainting Innocent Addresses

In the following, we will show that even if a DeFi platform can *perfectly* ban TC users, the censorship will cause new security issues. Specifically, we discuss a tainting attack, where the adversary can leverage TC withdrawals to taint innocent addresses. This attack can cause the victim addresses to be blocked when interacting with the frontend of censoring DeFi platforms (cf. Fig. 13).
Attack Strategy. Given a victim address `addr`, the adversary performs the following steps to taint `addr` and block `addr`'s activities on censoring platforms.
- 1. The adversary deposits coins into a mixer pool (e.g., TC 0.1 `ETH` pool).
- 2. Upon withdrawing coins from the pool, the adversary assigns the victim address `addr` as the withdrawal address.
- 3. The victim address `addr` will receive the assets from the pool; therefore, `addr` will be banned when interacting with censoring DeFi platforms.

Attack Cost. The attack cost is affected by the selected mixer pool. The cost of tainting an address equals the minimum denomination that the pool supports. The overall cost also increases linearly with the number of tainted addresses.
Attack Consequences. The tainting attack will lead the victim addresses to be banned when interacting with the censoring DeFi platforms' frontend. The ban could also cause utilization problems for DeFi users. For instance, on Aave,

blocked user addresses with active loans will not be able to access their borrowing position via the frontend and manage the position health to avoid being liquidated [21]. We provide an example to show how an adversary can benefit from tainting a user address on a censoring lending platform as follows.
- 1. Consider a user address `addr`, which supplies `ETH` and borrows `DAI` in Aave lending pool. The adversary performs the tainting attack against `addr`.
- 2. The adversary then leverages flash loans [24] to manipulate the price of `DAI`, which will cause the victim's borrowing position to become unhealthy.
- 3. As the victim is blocked by the Aave frontend and cannot access the position in time, the adversary can liquidate the unhealthy position to gain profits.

6.4 Bypassing Frontend-Level Censorship

In the following, we propose two methods to bypass frontend-level censorship.
Interacting with Smart Contracts via CLI. DeFi users can interact with the platform smart contracts through a CLI or by forking the platform project to create their own frontend interface. As shown in Fig. 12, in this way, there will be no third-party censorship, and user addresses will not be banned. However, this method might be beyond the technical knowledge of many DeFi users.
Leveraging Intermediary Addresses. Another method is to adopt a non-tainted address to interact with censoring DeFi platforms. To do so, users need to transfer their assets from their tainted addresses to non-tainted ones. For instance, we observe that a TC user transfers the withdrawn `ETH` to a non-tainted address via an intermediary address, to swap `ETH` to `renBTC` on Uniswap, i.e., $TC \xrightarrow{49.8\ \texttt{ETH}} \texttt{addr}_0 \xrightarrow{25.3\ \texttt{ETH}} \texttt{addr}_1 \xrightarrow{16.5\ \texttt{ETH}} \texttt{addr}_2 \xrightarrow{11.97\ \texttt{ETH}} Uniswap \xrightarrow{0.94\ \texttt{renBTC}}$. In this way, the non-tainted address $\texttt{addr}_2$ is not blocked by Uniswap.

7 Related Work

Blockchain Censorship. Moser et al. [11] discuss how transaction blacklisting would change the Bitcoin ecosystem and how it can remain effective in the presence of privacy-preserving blockchains. Kolachala et al. [8] investigate the blacklisting technique to combat money laundering, and point out that there are unanswered questions and challenges with regard to its enforcement.
Money Laundering on Blockchains. Wang et al. [20] investigate how users leverage ZKP mixers, e.g., TC and Typhoon.Network to launder money. Wang et al. also propose heuristics to link mixer deposit and withdrawal addresses, which can be used to trace mixer users' coin flow. Zhou et al. [24] indicate that DeFi attackers can receive their source of funds from mixers to launch attacks. Their results show that 55 (21%) and 12 (4.6%) of the 181 attack funds originate from the ZKP mixers on ETH and Binance Smart Chain, respectively.
Blockchain DoS Attacks. Li et al. [10] propose a series of low-cost DoS attacks named DETER, which leverages Ethereum clients' vulnerability in managing

unconfirmed transactions. DETER can disable a remote Ethereum node's mempool and deny the critical downstream services in mining and transaction propagation. Perez et al. [13] present a DoS attack, called Resource Exhaustion Attack, which systematically exploits the imperfections of the Ethereum metering mechanism to generate low-throughput contracts.

8 Conclusion

This paper studies the security implications of blockchain transaction censorship. Specifically, we show that miners or validators can execute the transaction to censor whether blacklisted addresses are called. This additional execution requirement enables an attack whereby an adversary could deliberately DoS a censoring miner or validator through crafting numerous tainted and complicated transactions. Our analysis shows that the attack comes at zero transaction fees when all miners or validators adopt censoring. Moreover, we find that a censoring FaaS might also suffer from such an attack. Furthermore, we show that current DeFi platforms' censorship is at the frontend level, and users can efficiently bypass the censorship using CLI or intermediary addresses. We hope our work can engender further research into more secure solutions for blockchain censorship.

Acknowledgments. We thank Pascal Berrang and anonymous reviewers from MARBLE 2023 for providing valuable comments which helped us to strengthen the paper. We are moreover grateful to Nimiq and SwissBorg SA for partially funding this work. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Nimiq and SwissBorg SA.

References

1. Tornado cash. www.tornado.cash/, before August 8th, 2022
2. Aave. Address screening. www.docs.aave.com/faq/#address-screening
3. Chainalysis. Understanding tornado cash, its sanctions implications, and key compliance questions. www.blog.chainalysis.com/reports/tornado-cash-sanctions-challenges/
4. Daian, P., Goldfeder, S., Kell, T., Li, Y., Zhao, X., Bentov, I., Breidenbach, L., Juels, A.: Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In: 2020 IEEE Symposium on Security and Privacy (SP), pp. 910–927. IEEE (2020)
5. dydx. Tornado outage. www.dydx.exchange/blog/tornado-outage
6. Jerry, B., and Van Valkenburgh, P.: Analysis: What is and what is not a sanctionable entity in the tornado cash case (2022)
7. Kilbourn, Q.: Order flow, auctions and centralisation. In: The Science of Blockchain Conference (2022)
8. Kolachala, K., Simsek, E., Ababneh, M., Vishwanathan, R.: Sok: Money laundering in cryptocurrencies. In: The 16th International Conference on Availability, Reliability and Security, pp. 1–10 (2021)

9. Duc V Le, D.V., Gervais, A.: Amr: Autonomous coin mixer with privacy preserving reward distribution. In: Advances in Financial Technologies (AFT'21) (2021)
10. Li, K., Wang, Y., Tang, Y.: Deter: Denial of ethereum txpool services. In: Proceedings of the 2021 ACM CCS, pp. 1645–1667 (2021)
11. Möser, M., Narayanan, A.: Effective cryptocurrency regulation through blacklisting. Preprint (2019)
12. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
13. Perez, D., Livshits, B.: Broken metre: Attacking resource metering in evm. In: Proceedings of the 27th NDSS. Internet Society (2020)
14. Qin, K., Zhou, L., Gervais, A.: Quantifying blockchain extractable value: How dark is the forest? In: IEEE Symposium on Security and Privacy (2022)
15. Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E. and Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: Symposium on Security and Privacy, pp. 459–474. IEEE (2014)
16. Tom, E.: Letter to treasury secretary yellen regarding the unprecedented sanctioning of tornado cash (2022). www.twitter.com/RepTomEmmer/status/1562084891247902721
17. U.S. Department of the Treasury. Cyber-related sanctions (2022). www.home.treasury.gov/taxonomy/term/1546
18. U.S. Department of the treasury. U.S. treasury sanctions notorious virtual currency mixer tornado cash (2022). www.home.treasury.gov/news/press-releases/jy0916
19. Uniswap. Address screening guide (2022). www.support.uniswap.org/hc/en-us/articles/8671777747597-Address-Screening-Guide
20. Wang, Z., Chaliasos, S., Qin, K., Zhou, L., Gao, L., Berrang, P., Livshits, B., Gervais, A.: On how zero-knowledge proof blockchain mixers improve, and worsen user privacy. In: Proceedings of the ACM Web Conference 2023, pp. 2022–2032 (2023)
21. Wang, Z., Qin, K., Minh, D.V., Gervais, A.: Speculative multipliers on defi: Quantifying on-chain leverage risks. In: Financial Cryptography and Data Security: 26th International Conference. FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers, pp. 38–56. Springer, Grenada (2022)
22. Weintraub, B., Torres, C.F., Nita-Rotaru, C., State, R.: A flash (bot) in the pan: Measuring maximal extractable value in private pools. In: Proceedings of the 22nd ACM Internet Measurement Conference (2022)
23. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger
24. Zhou, L., Xiong, X., Ernstberger, J., Chaliasos, S., Wang, Z., Wang, Y., Qin, K., Wattenhofer, R., Song, D., Gervais, A.: Sok: Decentralized finance (defi) incidents. arXiv:2208.13035 (2022)



An Automated Market Maker Minimizing Loss-Versus-Rebalancing

Conor McMenamin[1(✉)], Vanesa Daza[1,2], and Bruno Mazorra[1]

[1] Department of Information and Communication Technologies, Universitat Pompeu Fabra, Barcelona, Spain
{Conor.McMenamin,Vanesa.Daza,Bruno.Mazorra}@upf.edu
[2] CYBERCAT - Center for Cybersecurity Research of Catalonia, Tarragona, Spain

Abstract. The always-available liquidity of automated market makers (AMMs) has been one of the most important catalysts in early cryptocurrency adoption. However, it has become increasingly evident that AMMs in their current form are not viable investment options for passive liquidity providers. This is large part due to the cost incurred by AMMs providing stale prices to arbitrageurs against external market prices, formalized as loss-versus-rebalancing (LVR) (Milionis et al. 2022). In this paper, we present Diamond, an automated market making protocol that aligns the incentives of liquidity providers and block producers in the protocol-level retention of LVR. In Diamond, block producers effectively auction the right to capture any arbitrage that exists between the external market price of a Diamond pool, and the price of the pool itself. The proceeds of these auctions are shared by the Diamond pool and block producer in a way that is proven to remain incentive compatible for the block producer. Given the participation of competing arbitrageurs to capture LVR, LVR is minimized in Diamond. We formally prove this result, and detail an implementation of Diamond. We also provide comparative simulations of Diamond to relevant benchmarks, further evidencing the LVR-protection capabilities of Diamond. With this new protection, passive liquidity provision on blockchains can become rationally viable, beckoning a new age for decentralized finance.

1 Introduction

CFMMs such as Uniswap [17] have emerged as the dominant class of AMM protocols. CFMMs offer several key advantages for decentralized liquidity provision. They are efficient computationally, have minimal storage needs, matching computations can be done quickly, and liquidity providers can be passive. Thus, CFMMs are uniquely suited to the severely computation- and storage-constrained environment of blockchains.

Unfortunately, the benefits of CFMMs are not without significant costs. One of these costs is definitively formalized in [14] as *loss-versus-rebalancing* (LVR). It is proved that as the underlying price of a swap moves around in real-time, the discrete-time progression of AMMs leave arbitrage opportunities against

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
P. Pardalos et al. (Eds.): MARBLE 2023, 2024.
https://doi.org/10.1007/978-3-031-48731-6_6

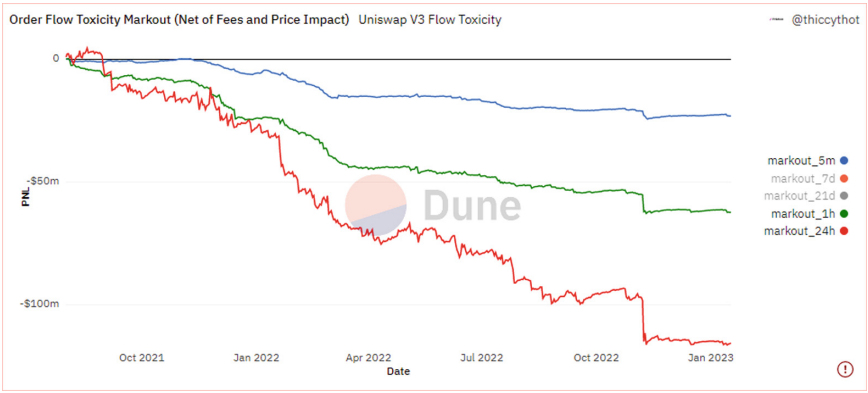


Fig. 1. Toxicity of Uniswap V3 Order Flow [16]. This graph aggregates the PnL (*toxicity*) of all trades on the Uniswap V3 WETH/USDC pool, measuring PnL of each order after 5 minutes, 1 h, and 1 day. These are typical time periods within which arbitrageurs close their positions against external markets. This demonstrates the losses being incurred in existing state-of-the-art DEX protocols are significant, consistent, and unsustainable; toxic.

the AMM. In centralized finance, market makers typically adjust to new price information before trading. This comes at a considerable cost to AMMs (for CFMMs, [14] derives the cost to be quadratic in realized moves), with similar costs for AMMs derived quantitatively in [6,15], and presented in Fig. 1.

These costs are being realized by liquidity providers in current AMM protocols. All of these factors point towards unsatisfactory protocol design, and a dire need for an LVR-resistant automated market maker. In this paper, we provide Diamond, an AMM protocol which formally protects against LVR.

1.1 Our Contribution

We present Diamond, an AMM protocol which isolates the LVR being captured from a Diamond liquidity pool, and forces some percentage of these LVR proceeds to be returned to the pool. As in typical CFMMs, Diamond pools are defined with respect to two tokens x and y. At any given time, the pool has reserves of R_x and R_y of both tokens, and some pool pricing function[1] $PPF(R_x, R_y)$. We demonstrate our results using the well-studied Uniswap V2 pricing function of $PPF(R_x, R_y) = \frac{R_x}{R_y}$.

In Diamond, block producers are able to capture the block LVR of a Diamond pool themselves or auction this right among a set of competing arbitrageurs. In both cases, the block producer revenue approximates the arbitrage revenue. Therefore, block producers are not treated as traditional arbitrageurs, but rather as players with effective arbitrage capabilities due to their unique position in blockchain protocols.

[1] See Eq. 1 for a full description of pool pricing functions as used in this paper

For each Diamond pool, we introduce the concept of its *corresponding CFMM pool*. Given a Diamond pool with token reserves (R_x, R_y) and pricing function $PPF(R_x, R_y) = \frac{R_x}{R_y}$, the corresponding CFMM pool is the Uniswap V2 pool with reserves (R_x, R_y). If a block producer tries to move the price of the corresponding CFMM pool adding Υ_x tokens and removing Υ_y, the same price is achieved in the Diamond pool by adding $(1-\beta)\Upsilon_x$ tokens for some $\beta > 0$, with β the *LVR rebate parameter*. The block producer receives $(1-\beta)\Upsilon_y$. In our framework, it can be seen that $PPF(R_x+(1-\beta)\Upsilon_x, R_y-(1-\beta)\Upsilon_y) < PPF(R_x+\Upsilon_x, R_y-\Upsilon_y)$, which also holds in our example, as $\frac{R_x+(1-\beta)\Upsilon_x}{R_y-(1-\beta)\Upsilon_y} < \frac{R_x+\Upsilon_x}{R_y-\Upsilon_y}$. A further υ_y tokens are removed from the Diamond pool to move the reserves to the same price as the corresponding CFMM pool, with these tokens added to a *vault*.

Half of the tokens in the vault are then periodically converted into the other token (at any time, all tokens in the vault are of the same denomination) in one of the following ways:

1. An auction amongst arbitrageurs.
2. Converted every block by the block producer at the final pool price. If the block producer must buy $\frac{\eta}{2}$ tokens to convert the vault, the block producer must simultaneously sell $\frac{\eta}{2}$ futures which replicate the price of the token to the pool. These futures are then settled periodically, either by
   (a) Auctioning the $\frac{\eta}{2}$ tokens corresponding to the futures amongst competing arbitrageurs with the protocol paying/collecting the difference.
   (b) The use of a decentralized price oracle. In this paper, we consider the use of the settlement price of an on-chain frequent batch auction, such as that of [13], which is proven to settle at the external market price in expectancy.

Importantly, these auctions are not required for protocol liveness, and can be arbitrarily slow to settle. We prove that all of these conversion processes have 0 expectancy for the block producer or Diamond pool, and prove that the LVR of a Diamond pool is $(1-\beta)$ of the corresponding CFMM pool. Our implementation of Diamond isolates LVR arbitrageurs from normal users, using the fact that arbitrageurs are always bidding to capture LVR. Specifically, if an LVR opportunity exists at the start of the block, an arbitrageur will bid for it in addition to ordering normal user transactions, meaning the proceeds of a block producer are at least the realized LVR, with LVR corresponding to the difference between the start- and end-states of an AMM in a given block. This ensures the protections of Diamond can be provided in practice while providing at least the same trading experience for normal users. Non-arbitrageur orders in a Diamond pool can be performed identically to orders in the corresponding CFMM pool after an arbitrageur has accepted to interact with the pool through a special arbitrageur-only transaction. Although this means user orders may remain exposed to the front-running, back-running and sandwich attacks of corresponding CFMMs, the LVR retention of Diamond pools should result in improved liquidity and reduced fees for users.

We discuss practical considerations for implementing Diamond, including decreasing the LVR rebate parameter, potentially to 0, during periods of protocol inactivity until transactions are processed, after which the parameter should be reset. This ensures the protocol continues to process user transactions, which becomes necessary when arbitrageurs are not actively extracting LVR. If arbitrageurs are not arbitraging the pool for even small LVR rebate parameters, it makes sense to allow transactions to be processed as if no LVR was possible. In this case, Diamond pools perform identically to corresponding CFMM pools. However, if/when arbitrageurs begin to compete for LVR, we expect LVR rebate parameters to remain high.

We present a series of experiments in Sect. 7 which isolate the benefits of Diamond. We compare a Diamond pool to its corresponding Uniswap V2 pool, as well as the strategy of holding the starting reserves of both tokens, demonstrating the power of Diamond. We isolate the effects of price volatility, LVR rebate parameter, pool fees, and pool duration on a Diamond pool. Our experiments provide convincing evidence that the relative value of a Diamond pool to its corresponding Uniswap V2 pool is increasing in each of these variables. These experiments further evidence the limitations of current CFMMs, and the potential of Diamond.

1.2 Organization of the Paper

Section 2 analyzes previous work related to LVR in AMMs. Section 3 outlines the terminology used in the paper. Section 4 introduces the Diamond protocol. Section 5 proves the properties of Diamond. Section 6 describes how to implement the Diamond protocol, and practical considerations which should be made. Section 7 provides an analysis Diamond over multiple scenarios and parameters, including a comparison to various reference strategies. We conclude in Sect. 8.

2 Related Work

There are many papers on the theory and design of AMMs, with some of the most important including [1–4,14]. The only peer-reviewed AMM design claiming protection against LVR [12] is based on live price oracles. The AMM must receive the price of a swap before users can interact with the pool. Such sub-block time price data requires centralized sources which are prone to manipulation, or require the active participation of AMM representatives, a contradiction of the passive nature of AMMs and their liquidity providers. We see this as an unsatisfactory dependency for DeFi protocols.

Attempts to provide LVR protection without explicit use of oracles either use predictive fees for all players [8] and/or reduce liquidity for all players through more complex constant functions [5]. Charging all users higher fees to compensate for arbitrageur profits reduces the utility of the protocol for genuine users, as does a generalized liquidity reduction. In Diamond, we only reduce liquidity for arbitrageurs (which can also be seen as an increased arbitrageur-specific fee),

providing at least the same user experience for typical users as existing AMMs without LVR protection.

A recent proposed solution to LVR published in a blog-post [10] termed *MEV-capturing AMMs* (McAMMs) considers auctioning off the first transaction/series of transaction in an AMM among arbitrageurs, with auction revenue paid in some form to the protocol. Two important benefits of Diamond compared to the proposed McAMMs are the capturing of realized LVR in Diamond as opposed to predicted LVR in McAMMs, and decentralized access to Diamond compared to a single point of failure in McAMMs.

In McAMMs, bidders are required to predict upcoming movements in the AMM. Bidders with large orders to execute over the period (e.g. private price information, private order flow, etc.) have informational advantages over other bidders. Knowing the difference between expected LVR excluding this private information vs. true expected LVR allows the bidder to inflict more LVR on the AMM than is paid for. As this results in better execution for the winner's orders, this may result in more private order flow, which exacerbates this effect. Diamond extracts a constant percentage of the true LVR, regardless of private information. McAMMs also centralize (first) access control to the winning bidder. If this bidder fails to respond or is censored, user access to the protocol is prohibited/more expensive. Diamond is fully decentralized, incentive compatible and can be programmed to effectively remove LVR in expectancy. Future McAMM design improvements based on sub-block time auctions are upper-bounded by the current protection provided by Diamond.

3 Preliminaries

This section introduces the key terminology and definitions needed to understand LVR, the Diamond protocol, and the proceeding analysis. In this work we are concerned with a single swap between token x and token y. We use x and y subscripts when referring to quantities of the respective tokens. The external market price of a swap is denoted by ε, while pool prices and price functions are denoted using a lowercase p and uppercase P respectively. The price of a swap is quoted as the quantity of token x per token y.

In this work we treat the block producer and an arbitrageur paying for the right to execute transactions in a block as the same entity. This is because the the arbitrageur must have full block producer capabilities, and vice versa, with the payoff for the block producer equal to that of an arbitrageur under arbitrageur competition. For consistency, and to emphasize the arbitrage that is taking place in extracting LVR, we predominantly use the arbitrageur naming convention. That being said, it is important to remember that this arbitrageur has exclusive access to building the sub-block of Diamond transactions. Where necessary, we reiterate that it is the block producer who control the per-block set of Diamond transactions, and as such, the state of the Diamond protocol.

3.1 Constant Function Market Makers

A CFMM is characterized by *reserves* $(R_x, R_y) \in \mathbb{R}^2_+$ which describes the total amount of each token in the pool. The price of the pool is given by *pool price function* $PPF : \mathbb{R}^2_+ \to \mathbb{R}$ taking as input pool reserves (R_x, R_y). PPF has the following properties:

$$\begin{aligned}
&\text{(a) } PPF \text{ is everywhere differentiable, with } \frac{\partial PPF}{\partial R_x} > 0, \ \frac{\partial PPF}{\partial R_y} < 0. \\
&\text{(b) } \lim_{R_x \to 0} PPF = 0, \ \lim_{R_x \to \infty} PPF = \infty, \ \lim_{R_y \to 0} PPF = \infty, \ \lim_{R_y \to \infty} PPF = 0. \\
&\text{(c) If } PPF(R_x, R_y) = p, \text{ then } PPF(R_x + cp, R_y + c) = p, \ \forall c > 0.
\end{aligned} \tag{1}$$

These are typical properties of price functions. Property (a) states the price of y is increasing in the number of x tokens in the pool and decreasing in the number of y tokens. Property (b) can be interpreted as any pool price value is reachable for a fixed R_x, by changing the reserves of R_y, and vice versa. Property (c) states that adding reserves to a pool in a ratio corresponding to the current price of the pool does not change the price of the pool. These properties trivially hold for the Uniswap V2 price function of $\frac{R_x}{R_y}$, and importantly allow us to generalize our results to a wider class of CFMMs.

For a CFMM, the *feasible set of reserves* C is described by:

$$C = \{(R_x, R_y) \in \mathbb{R}^2_+ : PIF(R_x, R_y) = k\} \tag{2}$$

where $PIF : \mathbb{R}^2_+ \to \mathbb{R}$ is the pool invariant and $k \in \mathbb{R}$ is a constant. The pool is defined by a smart contract which allows any player to move the pool reserves from the current reserves $(R_{x,0}, R_{y,0}) \in C$ to any other reserves $(R_{x,1}, R_{y,1}) \in C$ if and only if the player provides the difference $(R_{x,1} - R_{x,0}, R_{y,1} - R_{y,0})$.

Whenever an arbitrageur interacts with the pool, say at time t with reserves $(R_{x,t}, R_{y,t})$, we assume as in [14] that the arbitrageur maximizes their profits by exploiting the difference between $PPF(R_{x,t}, R_{y,t})$ and the external market price at time t, denoted ε_t. To reason about this movement, we consider a *pool value function* $V : \mathbb{R}_+ \to \mathbb{R}$ defined by the optimization problem:

$$V(\varepsilon_t) = \min_{(R_x, R_y) \in \mathbb{R}^2_+} \varepsilon_t R_y + R_x, \text{ such that } PIF(R_x, R_y) = k \tag{3}$$

Given an arbitrageur interacts with the pool with external market price ε_t, the arbitrageur moves the pool reserves to the (R_x, R_y) satisfying $V(\varepsilon_t)$.

3.2 Loss-Versus-Rebalancing

LVR, and its prevention in AMMs is the primary focus of this paper. The formalization of LVR [14] has helped to illuminate one of the main costs of providing liquidity in CFMMs. The authors of [14] provide various synonyms to conceptualize LVR. In this paper, we use the opportunity cost of arbitraging the pool

against the external market price of the swap, which is proven to be equivalent to LVR in Corollary 1 of [14]. The LVR between two blocks B_t and B_{t+1} where the reserves of the AMM at the end of B_t are $(R_{x,t}, R_{y,t})$ and the external market price when creating block B_{t+1} is ε_{t+1} is:

$$R_{x,t} + R_{y,t}\varepsilon_{t+1} - V(\varepsilon_{t+1}) = (R_{x,t} - R_{x,t+1}) + (R_{y,t} - R_{y,t+1})\varepsilon_{t+1}. \quad (4)$$

As this is the amount being lost to arbitrageurs by the AMM, this is the quantity that needs to be minimized in order to provide LVR protection. In Diamond, this minimization is achieved.

3.3 Auctions

To reason about the incentive compatibility of parts of our protocol, we outline some basic auction theory results.

First-price-sealed-bid-auction: There is a finite set of players $\mathcal{I}$ and a single object for sale. Each bidder $i \in \mathcal{I}$ assigns a value of X_i to the object. Each X_i is a random variable that is independent and identically distributed on some interval $[0, V_{max}]$. The bidders know its realization x_i of X_i. We will assume that bidders are risk neutral, that they seek to maximize their expected payoff. Per auction, each player submit a bid b_i to the auctioneer. The player with the highest bid gets the object and pays the amount bid. In case of tie, the winner of the auction is chosen randomly. Therefore, the utility of a player $i \in \mathcal{I}$ is

$$u_i(b_i, b_{-i}) = \begin{cases} \frac{x_i - b_i}{m}, & \text{if } b_i = \max_i\{b_i\}, \\ 0, & \text{otherwise} \end{cases}$$

where $m = |\text{argmax}_i\{b_i\}|$. In our protocol, we have an amount of tokens z that will be auctioned. This object can be exchanged by all players at the external market price ε. In this scenario, we have the following lemma. Proofs are included in the Appendix

Lemma 1. *Let $\mathcal{I}$ be a set of players that can exchange at some market any amount of tokens x or y at the external market price ε. If an amount z of token y is auctioned in a first-price auction, then the maximum bid of any Nash equilibrium is at least $z\varepsilon$.*

4 Diamond

This section introduces the Diamond protocol. When the core protocol of Sect. 4.2 is run, some amount of tokens are removed from the pool and placed in a *vault*. These vault tokens are eventually re-added to the pool through a conversion protocol. Sections 4.3 and 4.4 detail two conversion protocols which can be run in conjunction with the core Diamond protocol. Which conversion protocol to use depends on the priorities of the protocol users, with a discussion of their trade-offs provided in Sect. 7, and represented graphically in Fig. 2. These trade-offs can be summarized as follows:

- The process of Sect. 4.3 forces the arbitrageur to immediately re-add the removed tokens to the Diamond pool, while ensuring the ratio of pool tokens equals the external market price. This ratio is achieved by simultaneously requiring the arbitrageur to engage in a futures contract tied to the pool price, with the arbitrageur taking the opposite side of the contract. These futures offset any incentive to manipulate the ratio of tokens. This results in a higher variance of portfolio value for both the Diamond pool and the arbitrageur. In return for this risk, this process ensures the pool liquidity is strictly increasing in expectancy every block, with the excess value (reduced LVR) retained by the vault immediately re-added to the pool. This process can be used in conjunction with a decentralized price oracle to ensure the only required participation of arbitrageurs is in arbitraging the pool (see process 2 in Sect. 4.3). It should be noted that these futures contracts have collateral requirements for the arbitrageur, which has additional opportunity costs for the arbitrageur.
- The process in Sect. 4.4 converts the vault tokens periodically. This can result in a large vault balance accruing between conversions, with this value taken from the pool. This means the quality (*depth*) of liquidity is decreasing between conversions, increasing the impact of orders. From the AMM's perspective, this process incurs less variance in the total value of tokens owned by the pool (see Fig. 2), and involves a more straightforward and well-studied use of an auction (compared to a trusted decentralized oracle). There is also no collateral requirement for the arbitrageur outside of the block in which the arbitrage occurs. [2]

Section 5 formalizes the properties of Diamond, culminating in Theorem 1, which states that Diamond can be parameterized to reduce LVR arbitrarily close to 0. It is important to note that Diamond is not a CFMM, but the rules for adjusting pool reserves are dependent on a CFMM.

4.1 Model Assumptions

We outline here the assumptions used when reasoning about Diamond. In keeping with the seminal analysis of [14], we borrow a subset of the assumptions therein, providing here a somewhat more generalized model.

1. External market prices follow a martingale process.
2. The risk-free rate is 0.
3. There exists a population of arbitrageurs able to frictionlessly trade at the external market price, who continuously monitor and periodically interact with AMM pools.
4. An optimal solution (R_x^*, R_y^*) to Eq. 3 exists for every external market price $\varepsilon \geq 0$.

[2] As the arbitrageur and block producer are interchangeable from Diamond's perspective, we see the requirement for the block producer/arbitrageur to provide collateral in a block controlled by the block producer as having negligible cost.

The use of futures contracts in one version of the Diamond protocol makes the risk-free rate an important consideration for implementations of Diamond. If the risk free rate is not 0, the profit or loss related to owning token futures vs. physical tokens must be considered. Analysis of a non-zero risk-free rate is beyond the scope of the thesis.

4.2 Core Protocol

We now describe the core Diamond protocol, which is run by all Diamond variations. A Diamond pool Φ is described by reserves (R_x, R_y), a pool pricing function *PPF*(), a pool invariant function *PIF*(), an *LVR rebate parameter* $\beta \in (0, 1)$, and *conversion frequency* $\tau \in \mathbb{N}$.

We define the *corresponding CFMM pool* of Φ, denoted *CFMM*(Φ), as the CFMM pool with reserves (R_x, R_y) whose feasible set is described by pool invariant function *PIF*() and pool constant $k = PIF(R_x, R_y)$. Conversely, Φ is the *corresponding Diamond pool* of *CFMM*(Φ). It is important to note that the mapping of Φ to *CFMM*(Φ) is only used to describe the state transitions of Φ, with *CFMM*(Φ) changing every time the Φ pool reserves change.

Consider pool reserves $(R_{x,0}, R_{y,0})$ in Φ at time $t = 0$ (start of a block), and an arbitrageur wishing to move the price of Φ at time $t = 1$ (end of the block) to $p_1 = \frac{R_{x,1}}{R_{y,1}} \neq \frac{R_{x,0}}{R_{y,0}}$. In Diamond, to interact with the pool at time $t = 0$, the arbitrageur must deposit some amount of collateral, $(C_x, C_y) \in \mathbb{R}^2_+$. This is termed the *pool unlock transaction*. After the pool unlock transaction, the arbitrageur can then execute arbitrarily many orders (on behalf of themselves or users) against Φ, exactly as the orders would be executed in *CFMM*(Φ), as long as for any intermediate reserve state $(R_{x,i}, R_{y,i})$ after an order, the following holds:

$$C_x \geq \beta(R_{x,0} - R_{x,i}) \text{ and } C_y \geq \beta(R_{y,0} - R_{y,i}). \tag{5}$$

For end of block pool reserves $(R_{x,1}, R_{y,1})$, WLOG let $\Upsilon_y = R_{y,1} - R_{y,0} > 0$, and $\Upsilon_x = R_{x,0} - R_{x,1} > 0$ (the executed orders net bought x from Φ, and net sold y to Φ). The protocol then removes $\beta\Upsilon_y$ tokens from Φ, sending them to the arbitrageur, and adds $\beta\Upsilon_x$ tokens to Φ, taking these tokens from C_x. After this, it can be seen that $PPF(R_{x,1} + \beta\Upsilon_x, R_{y,1} - \beta\Upsilon_y) < p_1$. To ensure the reserves correspond to a *PPF* equal to p_1, a further $\upsilon_x > 0$ tokens are removed such that:[3]

$$PPF(R_{x,1} + \beta\Upsilon_x - \upsilon_x, R_{y,1} - \beta\Upsilon_y) = p_1. \tag{6}$$

These υ_x tokens are added to the *vault* of Φ. Summarizing the transition from $t = 0$ to $t = 1$ from the arbitrageur's perspective, this is equivalent to:

1. Adding $(1 - \beta)\Upsilon_y$ tokens to Φ and removing $(1 - \beta)\Upsilon_x$ tokens from Φ.

[3] Achievable as a result of properties(a) and (b) of Eq. 1.

2. Adding $\upsilon_x > 0$ tokens to the Φ vault from the Φ pool such that $PPF(R_{x,0} - (1-\beta)\Upsilon_x - \upsilon_x, R_{y,0} + (1-\beta)\Upsilon_y) = p_1$. Note, this is with respect to starting reserves. [4]

If only a single arbitrageur order is executed on Φ apart from the pool unlock transaction, the arbitrageur receives Υ_x tokens from the order, and must repay $\beta\Upsilon_x$ as a result of the pool unlock transaction. Any other sequence of orders resulting in a net Υ_x tokens being removed from Φ is possible, but $\beta\Upsilon_x$ tokens must always be repaid to the pool by the arbitrageur. As the arbitrageur has full control over which orders are executed, such sequences of orders must be at least as profitable for the arbitrageur as the single arbitrage order sequence.[5]

Vault Rebalance

After the above process, let there be $(\upsilon_x, \upsilon_y) \in \mathbb{R}^2_+$ tokens in the vault of Φ. If $\upsilon_y\varepsilon_1 > \upsilon_x$, add $(\upsilon_x, \frac{\upsilon_x}{\varepsilon_1})$ tokens into Φ from the vault. Otherwise, add $(\upsilon_y\varepsilon_1, \upsilon_y)$ tokens into Φ from the vault. This is a *vault rebalance*.

Every τ blocks, after the vault rebalance, the protocol converts half of the tokens still in the vault of Φ (there can only be one token type in the vault after a vault rebalance) into the other token in Φ according to one of either conversion process 1 (Sect. 4.3) or 2 (Sect. 4.4). The goal of the conversion processes is to add the Diamond vault tokens back into the Diamond liquidity pool in a ratio corresponding to the ε, while preserving the value of the tokens to be added to the pool.

To understand why half of the tokens are converted, assume WLOG that there are υ_x tokens in the vault. Given an external market price ε, $\frac{\upsilon_x}{2}$ tokens can be exchanged for $\upsilon_y = \frac{\upsilon_x}{2}\frac{1}{\varepsilon}$ tokens, and vice versa. Both conversion processes are constructed to ensure the expected revenue of conversion is at least $\upsilon_y = \frac{\upsilon_x}{2}\frac{1}{\varepsilon}$. Therefore, after conversion, there are at least $\frac{\upsilon_x}{2}$ and $\upsilon_y = \frac{\upsilon_x}{2}\frac{1}{\varepsilon}$ tokens in the vault, with $\frac{\frac{\upsilon_x}{2}}{\upsilon_y} = \varepsilon$. The conversion processes then add the unconverted $\frac{\upsilon_x}{2}$ and converted υ_y tokens back into the Φ pool, with the ratio of these tokens approximating the external market price. Importantly, these tokens have value of at least the original vault tokens υ_x.

4.3 Per-block Conversion Versus Future Contracts

After every arbitrage, the arbitrageur converts η equal to half of the total tokens in the vault at the pool price p_c. Simultaneously, the arbitrageur sells to the pool η future contracts in the same token denomination at price p_c. Given the pool

[4] If $\Upsilon_y > 0$ tokens are to be removed from $CFMM(\Phi)$ with $\Upsilon_x > 0$ tokens to be added in order to achieve p_1, then $(1-\beta)\Upsilon_y$ tokens are removed from Φ and $(1-\beta)\Upsilon_x$ tokens are added to Φ, with a further $\upsilon_y > 0$ removed from Φ and added to the vault such that $PPF(R_{x,0} + (1-\beta)\Upsilon_x, R_{y,0} - (1-\beta)\Upsilon_y - \upsilon_y) = p_1$.

[5] An example of such a sequence is an arbitrage order to the external market price, followed by a sequence of order pairs, with each pair a user order, followed by an arbitrageur order back to the external market price. There are arbitrarily many other such sequences.

buys η future contracts at conversion price p_c, and the futures settle at price p_T, the protocol wins $\eta(p_T - p_c)$.

These future contracts are settled every τ blocks, with the net profit or loss being paid in both tokens, such that for a protocol settlement profit of PnL measured in token x and pool price p_T, the arbitrageur pays (s_x, s_y) with $PnL = s_x + s_y p_T$ and $s_x = s_y p_T$. These contracts can be settled in one of the following (non-exhaustive) ways:

1. Every τ blocks, an auction takes place to buy the offered tokens from the arbitrageurs who converted the pool at the prices at which the conversions took place. For a particular offer, a positive bid implies the converter lost/the pool won to the futures. In this case the converter gives the tokens to the auction winner, while the pool receives the winning auction bid. A negative bid implies the converter won/the pool lost to the futures. In this case, the converter must also give the tokens to the auction winner, while the pool must pay the absolute value of the winning bid to the auction winner.
2. Every τ blocks, a blockchain-based frequent batch auction takes place in the swap corresponding to the pool swap. The settlement price of the frequent batch auction is used as the price at which to settle the futures.

4.4 Periodic Conversion Auction

Every τ blocks, η equal to half of the tokens in the vault are auctioned to all players in the system, with bids denominated in the other pool token (bids for x tokens in the vault must be placed in y tokens, and vice versa). For winning bid b in token x (or token y), the resultant vault quantities described by $(s_x = b, s_y = \eta)$ (or $(s_x = \eta, s_y = b)$) are added to the pool reserves. In this case, unlike in Section 4.3, there are no restrictions placed on $\frac{s_x}{s_y}$.

5 Diamond Properties

This section outlines the key properties of Diamond. We first prove that both conversion process have at least 0 expectancy for the protocol.

Lemma 2. *Converting the vault every block vs. future contracts has expectancy of at least 0 for a Diamond pool.*

Lemma 3. *A periodic conversion auction has expectancy of at least 0 for a Diamond pool.*

Corollary 1. *Conversion has expectancy of at least 0 for a Diamond pool.*

With these results in hand, we now prove the main result of the paper. That is, the LVR of a Diamond pool is $(1 - \beta)$ of the corresponding CFMM pool.

Theorem 1. *For a CFMM pool CFMM(Φ) with LVR of $L > 0$, the LVR of Φ, the corresponding pool in Diamond, has expectancy of at most $(1 - \beta)L$.*

6 Implementation

We now detail an implementation of Diamond. The main focus of our implementation is ensuring user experience in a Diamond pool is not degraded compared to the corresponding CFMM pool. To this point, applying a β-discount on every Diamond pool trade is not viable. To avoid this, we only consider LVR on a per-block, and not a per-transaction basis. Given the transaction sequence, in/exclusion and priority auction capabilities of block producers, block producers can either capture the block LVR of a Diamond pool themselves, or auction this right among arbitrageurs.

From an implementation standpoint, who captures the LVR is not important, whether it is the producer themselves, or an arbitrageur who won an auction to bundle the transactions for inclusion in Diamond. As mentioned already, we assume these are the same entity, and as such it is the arbitrageur who must repay the LVR of a block. To enforce this, for a Diamond pool, we check the pool state in the first pool transaction each block and take escrow from the arbitrageur. This escrow is be used in part to pay the realized LVR of the block back to the pool. The first pool transaction also returns the collateral of the previous arbitrageur, minus the realized LVR (computable from the difference between the current pool state and the pool state at the beginning of the previous block). To ensure the collateral covers the realized LVR, each proceeding pool transaction verifies that the LVR implied by the pool state as a result of the transaction can be repaid by the deposited collateral.

We can reduce these collateral restrictions by allowing the arbitrageur to bundle transactions based on a *coincidence-of-wants* (CoWs) (matching buy and sell orders, as is done in CoWSwap [7]). This can effectively reduce the required collateral of the arbitrageur to 0. Given the assumed oversight capabilities of arbitrageurs is the same as that of block producers, we do not see collateral lock-up intra-block as a restriction, although solutions like CoWs are viable alternatives.

Our implementation is based on the following two assumptions:

1. An arbitrageur always sets the final state of a pool to the state which maximizes the LVR.
2. The block producer realizes net profits of at least the LVR corresponding to the final state of the pool, either as the arbitrageur themselves, or by auctioning the right to arbitrage amongst a set of competing arbitrageurs.

If the final price of the block is not the price maximizing LVR, the arbitrageur has ignored an arbitrage opportunity. The arbitrageur can always ignore non-arbitrageur transactions to realize the LVR, therefore, any additional included transactions must result in greater or equal utility for the arbitrageur than the LVR.

6.1 Core Protocol

The first transaction interacting with a Diamond pool Φ in every block is the *pool unlock transaction*, which deposits some collateral, $(C_x, C_y) \in \mathbb{R}^2_+$. Only one pool

unlock transaction is executed per pool per block. Every proceeding user order interacting with Φ in the block first verifies that the implied pool move stays within the bounds of Eq. 5. Non pool-unlock transactions are executed as they would be in the corresponding CFMM pool $CFMM(\Phi)$ (without a β discount on the amount of tokens that can be removed). These transactions are executed at prices implied by pushing along the $CFMM(\Phi)$ curve from the previous state, and as such, the ordering of transactions intra-block affects the execution price. If a Diamond transaction implies a move outside of the collateral bounds, it is not executed.

The next time a pool unlock transaction is submitted (in a proceeding block), given the final price of the preceding block was p_1, the actual amount of token x or y required to be added to the pool and vault (the $\beta\Upsilon$ and υ of the required token, as derived earlier in the section) is taken from the deposited escrow, with the remainder returned to the arbitrageur who deposited those tokens.

Remark 1. Setting the LVR rebate parameter too high can result in protocol censorship and/or liveness issues as certain arbitrageurs may not be equipped to frictionlessly arbitrage, and as such, repay the implied LVR to the protocol. To counteract this, the LVR rebate parameter should be reduced every block in which no transactions take place. As arbitrageurs are competing through the block producers to extract LVR from the pool, the LVR rebate parameter will eventually become low enough for block producers to include Diamond transactions. After transactions have been executed, the LVR rebate parameter should be reset to its initial value. Rigorous testing of initial values and decay curves are required for any choice of rebate parameter.

6.2 Conversion Protocols

The described implementations in this section assume the existence of a decentralized on-chain auction.[6]

6.2.1 Per-block Conversion Versus Futures Given per-block conversion (Sect. 4.3), further deposits from the arbitrageur are required to cover the token requirements of the conversion and collateralizing the futures. The conversions for a pool Φ resulting from transactions in a block take place in the next block a pool unlock transaction for Φ is called. Given a maximum expected percentage move over τ blocks of σ_T, and a conversion of λ_y tokens at price p, the arbitrageur collateral must be in quantities π_x and π_y such that if the arbitrageur is long the futures:

$$1.\ \pi_x + \pi_y \frac{p}{1+\sigma_T} \geq \lambda_y (p - \frac{p}{1+\sigma_T}),\ \text{and}\ 2.\ \frac{\pi_x}{\pi_y} = \frac{p}{1+\sigma_T}. \tag{7}$$

[6] First-price sealed-bid auctions can be implemented using a commit-reveal protocol. An example of such a protocol involves bidders hashing bids, committing these to the blockchain along with an over-collaterlization of the bid, with bids revealed when all bids have been committed.

If the arbitrageur is short the futures it must be that:

$$1.\ \pi_x + \pi_y p(1+\sigma_T) \geq \lambda_y p \sigma_T, \qquad \text{and } 2.\ \frac{\pi_x}{\pi_y} = p(1+\sigma_T). \tag{8}$$

The first requirement in both statements is for the arbitrageur's collateral to be worth more than the maximum expected loss. The second requirement states the collateral must be in the ratio of the pool for the maximum expected loss (which also ensures it is in the ratio of the pool for any other loss less than the maximum expected loss). This second requirement ensures the collateral can be added back into the pool when the futures are settled.

At settlement, if the futures settle in-the-money for the arbitrageur, tokens are removed from the pool in the ratio specified by the settlement price with total value equal to the loss incurred by the pool, and paid to the arbitrageur. If the futures settle out-of-the-money, tokens are added to the pool from the arbitrageur's collateral in the ratio specified by the settlement price with total value equal to the loss incurred by the arbitrageur. The remaining collateral is returned to the arbitrageur. The pool constant is adjusted to reflect the new balances.

Remark 2. As converting the vault does not affect pool availability, the auctions for converting the vault can be run sufficiently slowly so as to eliminate the risk of block producer censorship of the auction. We choose to not remove tokens from the pool to collateralize the futures as this reduces the available liquidity within the pool, which we see as an unnecessary reduction in benefit to users (which would likely translate to lower transaction fee revenue for the pool). For high volatility token pairs, τ should be chosen sufficiently small so as to not to risk pool liquidation.

If Diamond with conversion versus futures is run on a blockchain where the block producer is able to produce multiple blocks consecutively, this can have an adverse effect on incentives. Every time the vault is converted and tokens are re-added to the pool, the liquidity of the pool increases. A block producer with control over multiple blocks can move the pool price some of the way towards the maximal LVR price, convert the vault tokens (which has 0 expectancy from Lemma 2), increase the liquidity of the pool, then move the pool towards the maximal LVR price again in the proceeding block. This process results in a slight increase in value being extracted from the pool in expectancy compared to moving the pool price immediately to the price corresponding to maximal LVR. Although the effect on incentives is small, re-adding tokens from a conversion slowly/keeping the pool constant unchanged mitigates/removes this benefit for such block producers.

6.2.2 Periodic Conversion Auction Every τ blocks, η equal to half the tokens in the vault are auctioned off, with bids denominated in the other token. The winning bidder receives these η tokens. The winning bid, and the remaining η tokens in the vault, are re-added to the pool.

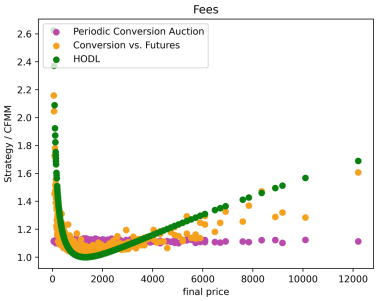


Fig. 2. .

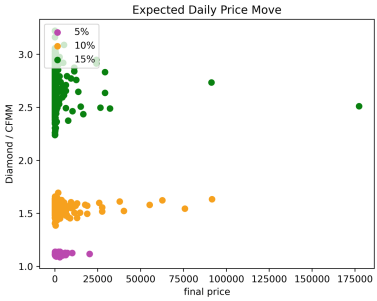


Fig. 3. .

7 Experimental Analysis

This section presents the results of several experiments, which can be reproduced using the following public repository [9]. The results provide further evidence of the performance potential of a Diamond pool versus various benchmarks. These experiments isolate the effect that different fees, conversion frequencies, daily price moves, LVR rebate parameters, and days in operation have on a Diamond pool. Each graph represents a series of random-walk simulations which were run, unless otherwise stated, with base parameters of:

- LVR rebate parameter: 0.95.
- Average daily price move: 5%.
- Conversion frequency: Once per day.
- Blocks per day: 10.
- Days per simulation: 365.
- Number of simulations per variable: 500.

Parameter Intuition. For a Diamond pool to be deployed, we expect the existence of at least one tradeable and liquid external market price. As such, many competing arbitrageurs should exist, keeping the LVR parameter close to 1. 5% is a typical daily move for our chosen token pair. Given a daily move of 5%, the number of blocks per day is not important, as the per block expected moves can be adjusted given the daily expected move. Given a simulator constraint of 5,000 moves per simulation, we chose 10 blocks per day for a year, as opposed to simulating Ethereum over 5,000 blocks (less than 1 day's worth of blocks), as the benefits of Diamond are more visible over a year than a day.

Each graph plots the final value of the Diamond Periodic Conversion Auction pool (unless otherwise stated) relative to the final value of the corresponding Uniswap V2 pool. The starting reserve values are \$100 m USDC and 76,336 ETH, for an ETH price of \$1,310, the approximate price and pool size of the Uniswap ETH/USDC pool at the time of simulation [17]. Figure 2 compares four strategies over the same random walks. Periodic Conversion Auction and

Conversion vs. Futures replicate the Diamond protocol given the respective conversion strategies (see Sect. 4). HODL (Hold-On-for-Dear-Life), measures the performance of holding the starting reserves until the end of the simulation. The final pool value of these three strategies are then taken as a fraction of the corresponding CFMM pool following that same random walk. Immediately we can see all three of these strategies outperform the CFMM strategy in all simulations (as a fraction of the CFMM pool value, all other strategies are greater than 1), except at the initial price of 1310, where HODL and CFMM are equal, as expected.

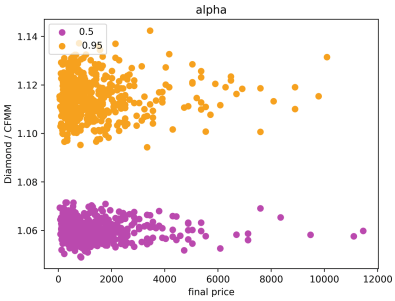


Fig. 4. .

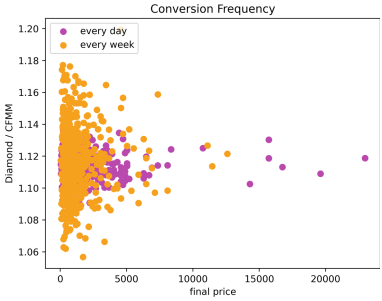


Fig. 5. .

The Diamond pools outperform HODL in a range around the starting price, as Diamond pools initially retain the tokens increasing in value (selling them eventually), which performs better than HODL when the price reverts. HODL performs better in tail scenarios as all other protocols consistently sell the token increasing in value on these paths. Note Periodic Conversion slightly outperforms Conversion vs. Futures when finishing close to the initial price, while slightly underperforming at the tails. This is because of the futures exposure. Although these futures have no expectancy for the protocol, they increase the variance of the Conversion versus Futures strategy, outperforming when price changes have momentum, while underperforming when price changes revert.

Figure 3 identifies a positive relationship between the volatility of the price and the out-performance of the Diamond pool over its corresponding CFMM pool. This is in line with the results of [14] where it is proved LVR grows quadratically in volatility. Figure 4 demonstrates that, as expected, a higher LVR rebate parameter β retains more value for the Diamond pool.

Figure 5 shows that higher conversion frequency (1 day) has less variance for the pool value (in this experiment once per day conversion has mean 1.1149 and standard deviation 0.0083 while once per week conversion has mean 1.1146 and standard deviation 0.0239). This highlights an important trade-off for protocol deployment and LPs. Although lower variance corresponding to more frequent

conversion auctions is desirable, more frequent auctions may centralize the players participating in the auctions due to technology requirements. This would weaken the competition guarantees needed to ensure that the auction settles at the true price in expectancy.

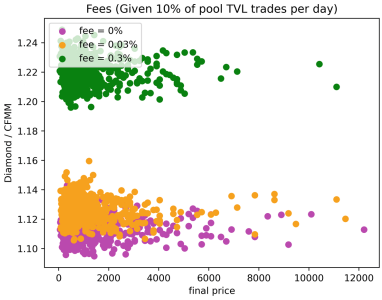


Fig. 6. .

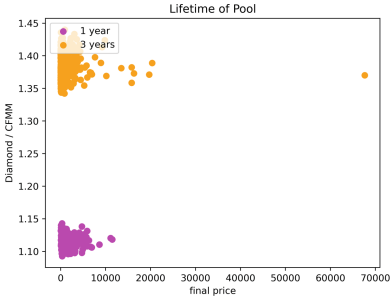


Fig. 7. .

Figure 6 compares Diamond to the CFMM pool under the specified fee structures (data-points corresponding to a particular fee apply the fee to both the Uniswap pool and the Diamond pool) assuming 10% of the total value locked in each pool trades daily. The compounding effect of Diamond's LVR rebates with the fee income every block result in a significant out-performance of the Diamond protocol as fees increase. This observation implies that given the LVR protection provided by Diamond, protocol fees can be reduced significantly for users, providing a further catalyst for a DeFi revival. Figure 7 demonstrates that the longer Diamond is run, the greater the out-performance of the Diamond pool versus its corresponding CFMM pool.

8 Conclusion

We present Diamond, an AMM protocol which provably protects against LVR. The described implementation of Diamond stands as a generic template to address LVR in any CFMM. The experimental results of Sect. 7 provide strong evidence in support of the LVR protection of Diamond, complementing the formal results of Sect. 5. It is likely that block producers will be required to charge certain users more transaction fees to participate in Diamond pools to compensate for this LVR rebate, with informed users being charged more for block inclusion than uninformed users. As some or all of these proceeds are paid to the pool with these proceeds coming from informed users, we see this as a desirable outcome.

Acknowledgements. We thank the reviewers for their detailed and insightful reviews, as well as Stefanos Leonardos for his guidance in preparing this camera-ready

version. This paper is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 814284, and is supported by the AEI-PID2021-128521OB-I00 grant of the Spanish Ministry of Science and Innovation.

A Proofs

Lemma 1. *Let $\mathcal{I}$ be a set of players that can exchange at some market any amount of tokens x or y at the external market price ε. If an amount z of token y is auctioned in a first-price auction, then the maximum bid of any Nash equilibrium is at least $z\varepsilon$.*

Proof. By construction, we have that the support of X_i is lower bounded by $z\varepsilon$. Therefore, in a second-price auction, in equilibrium, each player will bid at least, $z\varepsilon$. Using the revenue equivalence theorem [11], we deduce that the revenue of the seller is at least $z\varepsilon$ obtaining the result.

Lemma 2. *Converting the vault every block vs. future contracts has expectancy of at least 0 for a Diamond pool.*

Proof. Consider a conversion of η tokens which takes place at time 0. Let the conversion be done at some price p_c, while the external market price is ε_0. WLOG let the protocol be selling η y tokens in the conversion, and as such, buying η y token futures at price p_c. The token sells have expectancy $\eta(p_c - \varepsilon_0)$. For the strategy to have at least 0 expectancy, we need the futures settlement to have expectancy of at least $\eta(\varepsilon_0 - p_c)$. In Section 4.3, two versions of this strategy were outlined. We consider both here. In both sub-proofs, we use the assumption that the risk-free rate is 0, which coupled with our martingale assumption for ε means the external market price at time t is such that $\mathbb{E}(\varepsilon_t) = \varepsilon_0$. We now consider the two options for settling futures outlined in Section 4.3

Option 1: Settle futures by auctioning tokens at the original converted price. The arbitrageur who converted tokens for the pool at price p_c must auction off the tokens at price p_c. Let the auction happen at time t, with external market price at that time of ε_t. Notice that what is actually being sold is the right, and obligation, to buy η tokens at price p_c. This has value $\eta(\varepsilon_t - p_c)$, which can be negative. As negative bids are paid to the auction winner by the protocol, and positive bids are paid to the protocol, we are able to apply Lemma 1. As such, the winning bid is at least $\eta(\varepsilon_t - p_c)$, which has expectancy of at least

$$\mathbb{E}(\eta(\varepsilon_t - p_c)) = \eta(\mathbb{E}(\varepsilon_t) - p_c) = \eta(\varepsilon_0 - p_c). \tag{9}$$

Thus the expectancy of owning the future for the protocol is at least $\eta(\varepsilon_0 - p_c)$, as required.

Option 2: Settle futures using frequent batch auction settlement price. For a swap with external market price ε_t at time t, a batch auction in this swap settles at ε_t in expectancy (Theorem 5.1 in [13]). Thus the futures owned by the protocol have expectancy

$$\mathbb{E}(\eta(\varepsilon_t - p_c)) = \eta(\mathbb{E}(\varepsilon_t) - p_c) = \eta(\varepsilon_0 - p_c). \tag{10}$$

Lemma 3. *A periodic conversion auction has expectancy of at least 0 for a Diamond pool.*

Proof. Consider a Diamond pool Φ with vault containing 2η tokens. WLOG let these be of token y. Therefore the pool must sell η tokens at the external market price to balance the vault. Let the conversion auction accept bids at time t, at which point the external market price is ε_t. For the auction to have expectancy of at least 0, we require the winning bid to be at least $\eta\varepsilon_t$. The result follows from Lemma 1.

Theorem 1. *For a CFMM pool CFMM(Φ) with LVR of $L > 0$, the LVR of Φ, the corresponding pool in Diamond, has expectancy of at most $(1-\beta)L$.*

Proof. To see this, we first know that for $CFMM(\Phi)$ at time t with reserves $(R_{x,t}, R_{y,t})$, LVR corresponds to the optimal solution $(R^*_{x,t+1}, R^*_{y,t+1})$ with external market price ε_{t+1} which maximizes:

$$(R_{x,t+1} - R_{x,t}) + (R_{y,t+1} - R_{y,t})\varepsilon_{t+1}. \tag{11}$$

Let this quantity be

$$L = (R^*_{x,t+1} - R_{x,t}) + (R^*_{y,t+1} - R_{y,t})\varepsilon_{t+1}. \tag{12}$$

In Diamond, a player trying to move the reserves of Φ to $(R'_{x,t+1}, R'_{y,t+1})$ only receives $(1-\beta)(R'_{x,t+1} - R_{x,t})$ while giving $(1-\beta)(R'_{y,t+1} - R_{y,t})$ to Φ. Thus, an arbitrageur wants to find the values of $(R'_{x,t+1}, R'_{y,t+1})$ that maximize:

$$(1-\beta)(R'_{x,t+1} - R_{x,t}) + (1-\beta)(R'_{y,t+1} - R_{y,t})\varepsilon_{t+1} + \mathbb{E}(\text{conversion}). \tag{13}$$

where $\mathbb{E}(\text{conversion})$ is the per-block amortized expectancy of the conversion operation for the arbitrageurs. From Lemma 1, we know $\mathbb{E}(\text{conversion}) \geq 0$ for Φ. This implies the arbitrageur's max gain is less than:

$$(1-\beta)(R'_{x,t+1} - R_{x,t}) + (1-\beta)(R'_{y,t+1} - R_{y,t})\varepsilon_{t+1}, \tag{14}$$

for the $(R'_{x,t+1}, R'_{y,t+1})$ maximizing Equation 13. From Equation 12, we know this has a maximum at $(R'_{x,t+1}, R'_{y,t+1}) = (R^*_{x,t+1}, R^*_{y,t+1})$. Therefore, the LVR of Φ is at most:

$$(1-\beta)((R^*_{x,t+1} - R_{x,t}) + (R^*_{y,t+1} - R_{y,t})\varepsilon_{t+1}) = (1-\beta)L. \tag{15}$$

References

1. Adams, H., Keefer, R., Salem, M., Zinsmeister, N., Robinson, D.: Uniswap V3 Core (2021). www.uniswap.org/whitepaper-v3.pdf
2. Adams, H., Zinsmeister, N., Robinson, D.: Uniswap V2 Core (2020). www.uniswap.org/whitepaper.pdf

3. Bartoletti, M., Chiang, J.H.Y., Lluch-Lafuente, A.: A theory of automated market makers in DeFi. In: Damiani, F., Dardha, O. (eds.) Coordination Models and Languages, pp. 168–187. Springer International Publishing, Cham (2021)
4. Bartoletti, M., Chiang, J.H.Y., Lluch-Lafuente, A.: Maximizing extractable value from automated market makers. In: Financial Cryptography and Data Security. Springer, Berlin, Heidelberg (2022)
5. Bichuch, M., Feinstein, Z.: Axioms for Automated Market Makers: a Mathematical Framework in FinTech and Decentralized Finance. www.arxiv.org/abs/2210.01227 (2022). Accessed 10 Feb. 2023
6. Capponi, A., Jia, R.: The Adoption of Blockchain-based Decentralized Exchanges (2021). www.arxiv.org/abs/2103.08842. Accessed 10 Feb. 2023
7. CoW Protocol. www.docs.cow.fi/. Accessed 11 Oct. 2022
8. Evans, A., Angeris, G., Chitra, T.: Optimal fees for geometric mean market makers. In: Bernhard, M., Bracciali, A., Gudgeon, L., Haines, T., Klages-Mundt, A., Matsuo, S., Perez, D., Sala, M., Werner, S. (eds.) Financial Cryptography and Data Security. FC 2021 International Workshops, pp. 65–79. Springer, Berlin, Heidelberg (2021)
9. Github (2022). www.github.com/The-CTra1n/LVR
10. Josojo: MEV capturing AMMs (2022). www.ethresear.ch/t/mev-capturing-amm-mcamm/13336. Accessed 10 Feb 2023
11. Krishna, V.: Auction Theory. Academic Press (2009)
12. Krishnamachari, B., Feng, Q., Grippo, E.: Dynamic automated market makers for decentralized cryptocurrency exchange. In: 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pp. 1–2 (2021). https://doi.org/10.1109/ICBC51069.2021.9461100
13. McMenamin, C., Daza, V., Fitzi, M., O'Donoghue, P.: FairTraDEX: A Decentralised Exchange Preventing Value Extraction. In: Proceedings of the 2022 ACM CCS Workshop on Decentralized Finance and Security, pp. 39–46. DeFi'22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3560832.3563439, www.doi.org/10.1145/3560832.3563439
14. Milionis, J., Moallemi, C.C., Roughgarden, T., Zhang, A.L.: Quantifying loss in automated market makers. In: Zhang, F., McCorry, P. (eds.) Proceedings of the 2022 ACM CCS Workshop on Decentralized Finance and Security. ACM (2022)
15. Park, A.: The Conceptual flaws of constant product automated market making. ERN: Other Microeconomics: General Equilibrium & Disequilibrium Models of Financial Markets (2021)
16. @thiccythot: www.dune.com/thiccythot/uniswap-markouts. Accessed 10 Feb. 2023
17. Uniswap (2022). www.app.uniswap.org/. Accessed 11 Feb. 2022



Profit Lag and Alternate Network Mining

Cyril Grunspan[1(✉)] and Ricardo Pérez-Marco[2]

[1] Research Center, Léonard de Vinci Pôle University, 92 916 Paris La Défense, France
cyril.grunspan@devinci.fr
[2] CNRS, IMJ-PRG, Université Paris-Diderot, Paris, France

Abstract. For a mining strategy we define "profit lag" as the minimum time it takes to be profitable after that moment. We compute closed forms for the profit lag and the revenue ratio for the strategies "selfish mining" and "intermittent selfish mining". This corroborates prior numerical simulations and provides further elucidation regarding the issue of profitability as discussed in the existing literature. We also study mining pairs of PoW cryptocurrencies, often coming from a fork, with the same mining algorithm. This represents a vector of attack that can be exploited using the "alternate network mining" strategy that we define. We compute closed forms for the profit lag and the revenue ratio for this strategy that is more profitable than selfish mining and intermittent selfish mining. It is also harder to counter since it does not rely on a flaw in the difficulty adjustment formula that is the reason for profitability of the other strategies.

Keywords: Bitcoin · Proof-of-work · Selfish mining · Intermittent selfish mining · Smart mining

1 Introduction

1.1 Nakamoto Consensus

The founding paper "Bitcoin, a peer-to-peer electronic cash system" was announced at the end of October 2008 on a cryptography mailing list [22,23]. A first version of the software implementing the new protocol developed in the article was then released on January 2009 [24].

Bitcoin gradually achieved worldwide success and is today the cornerstone of the new crypto-economy valued at several hundreds of billions of dollars.

Two main reasons have contributed to its success. The decentralization of the network allows to transfer value on the internet from user to user without the assistance of a third party. Therefore, no central server, nor bank, nor jurisdiction can block payments [23]. A second reason is the invention of "smart-contracts" which made bitcoin the first programmable currency [1,30]. This allows to construct the "Lightning Network", on top of the Bitcoin network that offers instant and secure money transfer [6].

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
P. Pardalos et al. (Eds.): MARBLE 2023, 2024.
https://doi.org/10.1007/978-3-031-48731-6_7

Bitcoin consensus, sometimes referred today as Nakamoto Consensus, is based on the use of Proof of Work (PoW). PoW was originally invented to fight e-mail spam [7]. It was then used by H. Finney in the design of RPOW, the first cryptocurrency based on PoW, but was not decentralized. PoW plays a crucial role to secure Bitcoin protocol and in the minting process of new bitcoins. It combines computer security and classical probability theory. The main problem solved by Bitcoin and its PoW design is the prevention of double spend attacks without relying on a central server. Instead of searching for a flawless deterministic distributed consensus, Nakamoto's point is that, under reasonable conditions, the probability of success of such attacks is negligible [11,13,23,29] and economically non-profitable [17].

1.2 Mining Process

At any time, miners are working to build a new block from new transactions. This is achieved solving a laborious cryptographic puzzle which involves heavy computation and use of energy. Miners iterate the calculation of images of a cryptographic hash function of data from a new block by varying a "nonce" (and an "extra-nonce") until finding a result fulfilling certain criterion.

If successful, the miner broadcasts his discovery to the rest of the network which checks that the solution is legit. Then the new block is added to the previous known chain of blocks. The miner is then rewarded in newly minted bitcoins determined by the protocol and by the transaction fees of transactions included in the validated block. This sequence of blocks forms the "Bitcoin blockchain". This is a secured distributed ledger recording all past validated transactions [31].

1.3 Selfish Mining

On the early days, the community of bitcoiners met in different internet forums, among them bitcointalk.org, a forum originally created by Nakamoto in November 2009. In particular, they tried to understand in depth how Bitcoin works. Some people had doubts of the specific point in the protocol that requires miners to broadcast their blocks as soon as they are validated through the proof-of-work [27]. This fact allows the blockchain to record several thousand transactions every ten minutes in average. If a miner withholds secretly a block, he risks to lose the reward awarded to this block (coinbase and transaction fees) to a faster miner.

It is implicit in the founding paper that Nakamoto believed this to be the optimal strategy for miners [23]. Moreover, it is required in a decentralized protocol that the private economic interests of participants are in line with the protocol rules.

However, several deviant strategies were proposed in the bitcointalk.org forum and a seminal paper by Rosenfeld examined the problem in 2011 [27,28]. Then by the end of 2013, two articles showed that other alternative strategies than the honest one can be more profitable under suitable conditions [2,8].

By modeling the progression of the blockchain with a Markov chain, the authors showed that a certain deviant strategy, the so-called "selfish mining" is

more profitable in the long run than honest mining [8]. This is in particular the case when a mining pool detains slightly more than 33 % of the whole network hashpower.

The assumptions underlying the result, such as the cohesion of mining pools, were challenged. In particular, the fact that the model considers only two groups of miners and only one selfish mining pool in the network [3,9]. For example, one could imagine that some miners participating in a mining pool decide to selfish mine not only against honest miners but also against their own mining pool.

Using new martingale techniques and a rigorous profitability model, it was proved that the true reason for the profitability of selfish mining resides in a flaw in the difficulty adjustment formula [14]. An important result obtained by these new techniques is that without difficulty adjustment, the honest mining strategy is the most profitable one, hence vindicating Nakamoto's original belief. The flaw in the difficulty adjustment formula can be easily corrected. In the presence of a block-withholding attacker, the network, that does not track the production of orphan blocks, underestimates the real hash power deployed. The difficulty parameter does not reflect the exact mining activity in the network. Ultimately this makes the production of blocks easier than normal, which boosts miners income.

In particular, the selfish mining strategy only becomes profitable after a first difficulty adjustment, and after recouping all the cost employed reducing the difficulty (at the expense of regular honest mining income). At first, blocks are generated at a slower pace and all miners have their profitability reduced. Then after a first difficulty adjustment, blocks arrive faster and the selfish miner profits.

This theoretically explains why the strategy has probably never been implemented. No miner risks to mine at a loss for several weeks while assuming that mining conditions remain the same, i.e. one selfish miner and same overall hash power over time. Moreover, if a miner has substantial hashpower, then probably double spend attacks based on 51% hashrate dominance are easier to achieve by increasing the hashpower.

The original selfish mining strategy assumes also the non-realistic hypothesis that there is no arrival of new miners to the network attracted by the lower mining difficulty.

1.4 Smart Mining

As explained, the advantage of selfish mining is based on reducing the difficulty but continue to profit from block rewards, even before the first difficulty adjustment. Another idea presented consists in withdrawing temporarily from the network to lower the difficulty and then come back after a difficulty adjustment to take advantage of the lower hash rate mining [12]. The authors call such a strategy "smart mining" (sic). By considering the cost of mining (fixed cost and variable cost per time unit), it has been shown that this strategy can be more profitable than honest mining, even for low hashrates. In some sense,

this attack is more serious than the selfish mining strategy because it does not exploit a particular flaw of the difficulty adjustment formula.

1.5 Intermittent Selfish Mining

Another possible deviant strategy is the strategy of "Intermittent Selfish Mining" (ISM) that consists in alternating selfish and honest mining during consecutive difficulty periods. This was early discussed in social media and then numerically studied in [26]. The idea is to fully profit from the decrease of the difficulty but the downside is that the difficulty does not stabilize and recovers after each phase of honest mining.

1.6 Alternate Network Mining

In this article we introduce and study a strategy similar to "smart mining" that we name "Alternate Network Mining" (ANM). The difference is that when the miner withdraws from the network he goes on to mine in another network with similar mining algorithm. Clearly, if smart mining is profitable, "alternate network mining" is more profitable than "smart mining", and we prove that it is also more profitable than "selfish mining" and "intermittent selfish mining". It is also effective for low hashrates. Therefore, the existence of cryptocurrencies operating with identical PoW algorithms represents a vector of attack for both networks (this was already observed in [14]). Our study focus mainly on Bitcoin (BTC) and Bitcoin Cash (BCH), but can be adapted to other pairs of PoW cryptocurrencies sharing the same mining algorithm, such as BCH and Bitcoin Satoshi Vision (BSV).

1.7 Organization of This Article

In Sect. 2, we start briefly recalling the mathematics of Bitcoin mining and the profitability model for comparing mining strategies. We introduce our main notations and we define properly the notion of profit lag that is studied in this article. In Sect. 3, we review the selfish mining strategy and show the equivalence between the Markov chain approach and the martingale approach for the computation of profitabilities [8,14]. Also we review how to compute time before profit with both approaches. In Sect. 4, we turn to ISM strategy and compute in closed form the apparent hashrate of the strategy as well as the time before profit. These results are new since before only numerical simulations were available. Finally in Sect. 5, we compute the profitability of the ANM strategy. It is by far the most profitable mining strategy. It is also immediately and definitely profitable after just one difficulty adjustment. The present article is self-contained.

2 Modelization

2.1 Mining and Difficulty Adjustment Formula

From new transactions collected in the local database (the "mempool")[1] a miner builds a block B containing a trace of a previous block and a variable parameter (called a "nonce" but in fact an extra-nonce is also needed) until he finds a solution to the inequality $h(B) < \frac{1}{\delta}$ where δ is the difficulty of the hashing problem and h is a cryptographic hash function ($h = SHA_{256} \circ SHA_{256}$ for Bitcoin). This solution is the "proof-of-work" because it proves, under the usual assumptions on the pseudorandom properties of the hash function, that the miner has spent energy. For Bitcoin, the difficulty is adjusted every 2016 blocks, which corresponds on average to 2 weeks (there is actually a bias due partially to a minor bug in the implementation of the Difficulty Adjustment Algorithm [32]). For Bitcoin Cash and for Bitcoin SV, the difficulty is adjusted at each block using moving averages (exponential moving averages for BCH). The search for a good difficulty adjustment algorithm is itself a particularly active topic in the Bitcoin Cash community [20]. For previous Ethereum, the difficulty is adjusted at each block using the lapse for the discovery of the previous block. Ethereum difficulty adjustment algorithm was modified in June 2017 [4]. Since then, the adjustment took into account the production of special orphan blocks also called "uncles". These are orphan blocks that are not too far from the "parent block". Ethereum Classic did not implement this new difficulty adjustment formula, and is more vulnerable to selfish mining [10,19].

2.2 Notations

In the article, we use Nakamoto's notations from his founding paper. Thus, we denote by p (resp. $q = 1 - p$) the relative hash power of the honest miners (resp. the attacker). So, at any time, q is the probability that the attacker discovers a new block before the honest miners. We also denote by $N(t)$ (resp. $N'(t)$) the counting process of blocks validated by the honest miners (resp. the attacker) during a period of t seconds from an origin date 0. The hash function SHA_{256} is assumed to be a perfect pseudo-random hash function, the time it takes to find a block follows an exponential distribution and hence $N(t)$ (resp. $N'(t)$) is a Poisson process. Initially (before a difficulty adjustment) the parameter of the Poisson process $N(t)$ (resp. $N'(t)$) is $\frac{p}{\tau_0}$ (resp. $\frac{q}{\tau_0}$) with $\tau_0 = 600\,\mathrm{s}$. For $n \in \mathbb{N}$, we also denote by S_n (resp. S'_n) the time it takes for the honest miners (resp. the attacker) to mine n blocks, and $\tilde{S}_n$ the time it takes for the whole network to add n blocks to the official blockchain. The random variables S_n and S'_n follow Gamma distributions. The same occurs for $\tilde{S}_n$ when the attacker mines honestly [13]. The difficulty adjustment factor is denoted by δ. It is updated every $n_0 = 2016$ blocks. It is by definition the quotient between the time taken

[1] The "mempool" also known as a "memory pool" serves as a temporary storage space that holds pending transactions until they are added to a block. See for instance https://en.bitcoin.it/wiki/Vocabulary#Memory_pool.

to add the last n_0 blocks to the official blockchain and 2 weeks ($= n_0 \times 10$ minutes). Thus, $\tilde{S}_{n_0}$ represents a complete mining period of n_0 official blocks and at this date, the protocol proceeds with a new difficulty.

We use the notation γ for the connectivity of the attacker. In case of a competition between two blocks, one of which has been mined and published by the attacker, γ represents the fraction of honest miners mining on top of the attacker's block.

Last, the mean value of a block (coinbase and transaction fees) is denoted by b.

2.3 Profitability of a Mining Strategy

We consider a miner which is active over a long period compared to the average mining time of a block. What counts for his business is its Profit and Loss (PnL) per unit of time. We denote by $R(t)$ the total income of the miner between time 0 and time $t > 0$. Similarly, we denote by $C(t)$ the total cost he incurs during this period. Note that $C(t)$ is not restricted to direct mining costs but also includes all related expenditures (hardware costs, electricity costs, cooling costs, rental costs, personnel costs, internet connection costs, etc). So, his PnL is $PnL(t) = R(t) - C(t)$ and seeks to maximize $\frac{PnL(t)}{t} = \frac{R(t)}{t} - \frac{C(t)}{t}$ for $t \to \infty$. We assume that the mining cost is independent of the mining strategy. Indeed, whether the miner broadcasts his blocks or keeps them secret, it has no impact on its mining costs. Also, whether he mines a certain cryptocurrency or another with the same hash algorithm, this does not change the mining cost per unit of time. This last quantity essentially depends on electricity costs, price of his machines, salaries of employees, etc. In particular, when the miner mines at full regime, then the cost of mining per unit of time does not depend on the strategy. In this situation, the relevant quantity is the revenue ratio $\Gamma = \lim_{t \to +\infty} \frac{R(t)}{t}$. A strategy ξ is more profitable than a strategy ξ' if and only if its revenue ratio is greater: $\Gamma(\xi) \geq \Gamma(\xi')$ (see Corollary 1 below).

2.4 Attack Cycles

A strategy consists of attack cycles. The end of a cycle is determined by a *stopping time*. At the start of a cycle, the attacker and the honest miners have the same view of the official blockchain and mine on top of the same block. In general, though this is not mandatory, during the attack cycle, the attacker mines on a fork that he keeps secret.

Example 1. The sequence SSSHSHH corresponds to a particular attack cycle for the selfish mining strategy: the attacker first mines three blocks in a row that are kept secret (blocks "S"); then the honest miners mine one (block "H"); then the attacker mines another one (still secret); then the honest miners mine two blocks in a row and so the attacker decides to publish his entire fork because he only has a lead of 1 on the official blockchain. The attack cycle then ends. In this case, the attacker is victorious: all the blocks he has mined end up in the official blockchain.

The attacker iterates attack cycles. It follows that by noting R_i the miner's income after a i-th cycle and by T_i the duration time of this cycle, the revenue ratio of the strategy is equal to $\frac{\sum_{i=1}^{n} R_i}{\sum_{i=1}^{n} T_i}$ for $n \to +\infty$. This quantity converges to $\frac{\mathbb{E}[R]}{\mathbb{E}[T]}$ by the strong law of large numbers, provided that the duration time of a cycle attack T is integrable and non-zero. Likewise, assuming that costs are integrable, the cost of mining by time unit converges to $\frac{\mathbb{E}[C]}{\mathbb{E}[T]}$. Thus we can state the following theorem [14].

Theorem 1. *Let ξ and ξ' be two mining strategies. Let R and R' (resp. C and C') be the revenue (resp. cost) of the miner by attack cycle. We denote also by T and T' the duration times of attack cycles for ξ and ξ'. Then, ξ is more profitable than ξ' if and only if $\frac{\mathbb{E}[R_\xi]}{\mathbb{E}[T]} - \frac{\mathbb{E}[C_\xi]}{\mathbb{E}[T]} > \frac{\mathbb{E}[R_{\xi'}]}{\mathbb{E}[T]} - \frac{\mathbb{E}[C_{\xi'}]}{\mathbb{E}[T]}$.*

Corollary 1. *If moreover we assume that the cost of mining per unit of time of ξ and ξ' are equally distributed, then ξ is more profitable than ξ' if and only if $\frac{\mathbb{E}[R_\xi]}{\mathbb{E}[T]} > \frac{\mathbb{E}[R_{\xi'}]}{\mathbb{E}[T]}$.*

Example 2. The attack cycles for the honest strategy are simply the time taken by the whole network to discover a block. Consequently, when the miner mines honestly, his revenue ratio is $\frac{qb}{\tau_0}$.

2.5 Performant Strategy and Profit Lag

Definition 1. *A mining strategy is a performant strategy if its revenue ratio is greater than the revenue ratio of the honest strategy, i.e. $\frac{\mathbb{E}[R(T)]}{\mathbb{E}[T]} > \frac{qb}{\tau_0}$, where $R(T)$ is the revenue per attack cycle and T is the duration time of an attack cycle.*

For a performant strategy ξ, we have $\mathbb{E}[R(T)] \geq qb\frac{\mathbb{E}[T]}{\tau_0}$. So, at T-time, the miner is better off, on average, following strategy ξ that mining honestly. However, should the miner choose to continue with the ξ strategy, it is possible that they may incur losses if the balance is calculated at a date later than ξ. Nothing prevents having $\mathbb{E}[R(\tau)] < qb\frac{\mathbb{E}[\tau]}{\tau_0}$ for a certain stopping time $\tau > T$. This, indeed, happens for the ISM strategy. Below we define the notion of profit lag of a mining strategy.

Definition 2. *Let ξ be a mining strategy and τ an integrable stopping time.*

- *ξ is profitable at date τ if $\mathbb{E}[R(\tau)] \geq qb\frac{\mathbb{E}[\tau]}{\tau_0}$.*
- *ξ is definitely a performant strategy at date τ if $\mathbb{E}[R(\tau')] \geq qb\frac{\mathbb{E}[\tau']}{\tau_0}$ for any stopping time $\tau' > \tau$ a.s.*
- *The profit lag is the smallest stopping time τ with this property.*

This definition is sound since the infimum of stopping times is a stopping time. Note that if ξ is a performant strategy with duration time T for an attack cycle, then ξ is profitable at a date T. But in general, ξ is *not* definitely profitable at date T: the profit lag is in general longer than T. However we can prove that if ξ is a performant strategy then the profit lag is finite.

Proposition 1. *Let ξ be a performant mining strategy. Then, there exists τ a stopping time such that ξ is definitely a performant strategy at date τ.*

Proof. We keep the same notations as above. Let $X(t) = \mathbb{E}[R(t) - \frac{qbt}{\tau_0}]$ for $t \in \mathbb{R}_+$. Since ξ is a performant repetitive strategy, we have $X(T) > 0$ and

$$\mathrm{Inf}\{X(\tau)\,;\,\tau > 0 \text{ stopping time}\} = \mathrm{Inf}\{X(\tau)\,;\,\tau \text{ stopping time} \in [0,T]\} > -\infty$$

Let us denote by $m \in \mathbb{R}_-$ this quantity and let n be an integer with $n > \frac{|m|}{X(T)}$. Then, if $\tau > nT$ is a stopping time, we have:

$$X(\tau) = X(nT) + X(\tau - nT) \geq nX(T) + m > 0$$

Hence, we get the result.

3 Selfish Mining Revisited

Selfish mining strategy can be described by the stopping time which defines the end of an attack cycle.

Definition 3. *The end of an attack cycle for the selfish mining strategy is given by the stopping time* $T = \mathit{Inf}\left\{t > S_1 \,/\, N(t) = N'(t) - 1 + 2 \cdot 1_{S_1 < S_1'} + 2 \cdot 1_{S_1' < S_1 < S_2 < S_2'}\right\}$.

Example 3. In the case considered above, $S_1' < S_2' < S_3' < S_1 < S_4' < S_2 < S_3$, we have $1_{S_1 < S_1'} = 1_{S_1' < S_1 < S_2 < S_2'} = 0$ and $T = \mathrm{Inf}\left\{t > S_1 \,/\, N(t) = N'(t) - 1\right\} = S_3$.

We note also by L the number of blocks added to the official blockchain after an attack cycle. It is clear that $|N(T) - N'(T)| = 1$ so $L = \frac{N(T) + N'(T) + 1}{2}$. In the previous example, we have $N(T) = 3$, $N'(T) = 4$ and $L = 4$.

Proposition 2. *We have:*

$$\mathbb{E}[R] = \frac{(1+pq)(p-q)+pq}{p-q}\, qb - (1-\gamma)p^2 q\, b\ ,$$
$$\mathbb{E}[T] = \frac{(1+pq)(p-q)+pq}{p-q}\, \tau_0$$
$$\mathbb{E}[L] = \frac{1+p^2q+p-q}{p-q} b$$

Proof. The counting process $N(t)$ (resp. $N'(t)$) is a Poisson process. So, T (resp. T') is an integrable stopping time. Then the result follows from Doob stopping time theorem applied to the compensated martingales $N(t) - \alpha t$ (resp. $N'(t) - \alpha' t$) with $\alpha = \frac{p}{\tau_0}$ (resp. $\alpha' = \frac{q}{\tau_0}$) and to the finite stopping time $T \wedge t$ for a fixed $t > 0$. Then, we take the limit when $t \to \infty$ using the monotone convergence

theorem. We get as limits $\mathbb{E}[T]$, $\mathbb{E}[N(T)]$, $\mathbb{E}[N'(T)]$ and $\mathbb{E}[L]$. Finally, we also get $\mathbb{E}[R]$ by observing that $\mathbb{E}[R] = \mathbb{E}[N'(T)]b - (1-\gamma)p^2qb$. Indeed, the only situation in which $R \neq N'(T)$ occurs is when the attacker mines the first block, followed by two consecutive blocks mined by honest miners on top of each other and above the initial honest block.

We set the date $t = 0$ when the selfish miner starts his attack, at the beginning of a new period of 2016 official blocks, just after a difficulty adjustment.

Corollary 2. *Let δ be the first difficulty parameter. We have $\mathbb{E}[\delta] = \frac{p-q+pq(p-q)+pq}{p^2q+p-q}$ and $\mathbb{E}[\tilde{S}_{n_0}] = \mathbb{E}[\delta] \cdot n_0 \cdot \tau_0$*

Proof. Let ν be the number of cycles before we get a difficulty adjustment. By Wald theorem, we get $\mathbb{E}[L] \cdot \mathbb{E}[\nu] = n_0$ and $\mathbb{E}[\tilde{S}_{n_0}] = \mathbb{E}[\nu] \cdot \mathbb{E}[T]$. So, $\mathbb{E}[\delta] = \frac{\mathbb{E}[\tilde{S}_{n_0}]}{n_0\tau_0} = \frac{n_0\mathbb{E}[T]}{\mathbb{E}[L]\tau_0}$. Hence, we get the result using Proposition 2.

We can now calculate the apparent hashrate of the strategy which is defined as the fraction of blocks of the attacker in the official blockchain *on the long term*.

Corollary 3. *The apparent hashrate of the selfish mining strategy is*

$$q' = \frac{((1+pq)(p-q)+pq)q - (1-\gamma)p^2q(p-q)}{p^2q+p-q}$$

Proof. After a difficulty adjustment, the mining difficulty is divided by δ. So the parameters of the Poisson processes $N(t)$ and $N'(t)$ are each multiplied by δ and also the duration times of attack cycles are divided by δ. On the other hand, the revenue of the selfish miner remains constant after each attack cycle since the probability to be first to discover a new block remains the same. Therefore, his new revenue ratio after difficulty adjustment is $q'\frac{b}{\tau_0}$ with $q' = \left(\frac{\mathbb{E}[R]}{\mathbb{E}[T]} \cdot \delta\right) \cdot \frac{\tau_0}{b}$. Hence, we get the result using Proposition 2.

By rearranging terms, we observe that q' is nothing but the quantity R_{pool} from [8]. Also, we have proven the following proposition.

Proposition 3. *Before a difficulty adjustment, official blocks are discovered on average every period of $\delta \cdot \tau_0$. The revenue ratio of the selfish miner is $\frac{q'}{\delta}\frac{b}{\tau_0}$ (lower than $q\frac{b}{\tau_0}$). After a difficulty adjustment, officials blocks are discovered on average every period of τ_0. The revenue ratio of the selfish miner equal $q'\frac{b}{\tau_0}$ (which is now greater than $q\frac{b}{\tau_0}$).*

3.1 Previous State-Machine Approach Revisited

We can model the different states taken by the network with the help of a Markov chain (X_n) as was done in [8]. The set of states is $\mathbb{N} \cup \{0'\}$. The state $\{0'\}$ corresponds to the state when there is a competition between two blocks in the

network. The state $\{n\}$ corresponds to the state when the attacker has a lead of n blocks over the honest miners. In this framework, the duration time of an attack cycle is proportional to the first return time $\nu = \text{Inf}\,\{n \in \mathbb{N}^* \,/\, X_n = 0\}$. It turns out that the Markov chain is irreducible and all states are positive recurrent. It has naturally a stationary probability (π_n) and the computation shows that $\pi_0 = \frac{1-2q}{1-4q^2+2q^3}$ [8]. Therefore, a classical probability result on Markov chains shows that $\mathbb{E}[\nu] = \frac{1}{\pi_0}$ [5]. Then it is easy to check that

$$\mathbb{E}[\nu] = \frac{(1+pq)(p-q)+pq}{p-q}$$

We recover the formula obtained for $\mathbb{E}[T]$ in Proposition 2. Also, for r_{pool} and r_{others} defined in [8] we have

$$\frac{1}{r_{pool}+r_{others}} = \frac{1-4q^2+2q^3}{1-q-2q^2+q^3} = \mathbb{E}[\delta]$$

So we get the difficulty adjustment parameter given in Proposition 2. Finally, the revenue ratio before difficulty adjustment is $\frac{\mathbb{E}[R]}{\mathbb{E}[T]} = r_{pool} \cdot \frac{b}{\tau_0}$. The formula given in [8] for the apparent hashrate $q' = \frac{r_{pool}}{r_{pool}+r_{others}}$ is the same that we find: r_{pool} is the revenue ratio before difficulty adjustment and $\frac{1}{r_{pool}+r_{others}} > 1$ is the (mean) difficulty adjustment parameter δ. For the purposes of computation of the profitability there are two equivalent models: one with Poisson processes (as considered first by Satoshi Nakamoto in his founding paper and continued in [14]) and another with Markov chains [8]. They are not equivalent for other purposes, as for example to show the optimality of honest mining when there is no difficulty adjustment. Note also that there exists a third very effective combinatorial model that captures directly the long-term behavior of a deviant strategy. For the selfish mining strategy, stubborn strategy and trailing strategy, it uses the combinatorics of Dyck words and Catalan distribution [15,16,25].

3.2 Profit Lag

We recall that according to notations in Sect. 2.2, $\tilde{S}_{n_0}$ denotes the time used by the network to reach the first difficulty adjustment.

Proposition 4. *Set $t_0 = \frac{q\delta-q'}{q'-q} n_0\tau_0$. On average, the selfish mining strategy is not profitable before $\tilde{S}_{n_0} + t_0$ and is profitable after this date.*

Proof. We denote by $\Delta(t)$ the difference between the revenue of the selfish miner at date t and the revenue of honest mining. As we have seen, the revenue ratio of the selfish miner before the first difficulty adjustment is lower than the revenue ratio of the honest strategy. So, the selfish mining strategy cannot be profitable before this date. Moreover, using Proposition 3, we get: $\mathbb{E}[\Delta(\tilde{S}_{n_0})] = (\frac{q'}{\delta} - q)n_0\delta b = -(q\delta - q')n_0 b < 0$ and for $t > 0$, we have

$$\mathbb{E}[\Delta(\tilde{S}_{n_0}+t)] = (q'-q)\frac{t}{\tau_0}b + \mathbb{E}[\Delta(\tilde{S}_{n_0})] = (q'-q)\frac{t}{\tau_0}b - (q\delta-q')n_0 b = (q'-q)(t-t_0)\frac{b}{\tau_0}$$

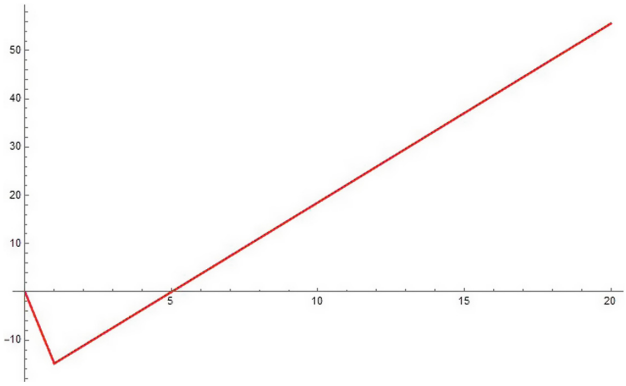


Fig. 1. Graph of $t \mapsto \mathbb{E}[R(t)] - qbt/\tau_0$ i.e., difference of average revenue between the selfish miner and what he would have won by mining honestly for $q = 0.1$ and $\gamma = 0.9$. X-axis: progression of the official blockchain, in difficulty adjustment units. Y-axis: revenue of the miner in coinbase units. The strategy is not definitely profitable before 5 difficulty adjustments. Note also that as soon as the strategy is profitable, it becomes *definitely* profitable.

Example 4. When $q = 0.1$ and $\gamma = 0.9$, the computation shows that we have exactly $\frac{q\delta - q'}{q' - q} = 4$. Therefore, the selfish mining strategy becomes definitely profitable at the date of the fifth difficulty adjustment which is roughly 10 weeks of mining (it is exactly 10.1839 weeks) as asserted in [14]. Before this date, the strategy is not profitable.

Note 1. In the introduction of [26] the authors claim that the previous computation from [14] is erroneous. However, in their simulations, they obtain $\mathbb{E}[\tilde{S}_{n_0}] = 15.29$ days which agrees, up to rounding, with the exact theoretical value from Corollary 2 which is 15.2872. And from here, it is not hard to get Proposition 4 as we did above. Note also that our theoretical exact results agree with existing selfish mining simulators that one can find online [21]. In fact, the authors in [26] do not explicitly "revisit" selfish mining but study numerically another related mining strategy called ISM. It is true that for the same choice of parameters as in the previous example ISM becomes profitable slightly before a second difficulty adjustment whereas at this date, SM is not. However, as we will see shortly, the strategy ISM is not definitely profitable before 13 difficulty adjustments. By comparison, SM is definitely profitable after 5 difficulty adjustments only. The rigorous notion of "profit lag" that we introduced above makes things clear. In the next section we give rigorous analysis of ISM with closed form formulas.

4 Intermittent Selfish Mining Strategy

This strategy is composed by attack cycles with two distinct phases. Each phase corresponds to a complete period of mining of n_0 official blocks until the next

difficulty adjustment. In the first phase, the attacker follows the selfish mining strategy. In the second phase, he mines honestly.

Proposition 5. *The mean duration time of the first (resp. second) phase is* $n_0\,\tau_0\,\delta$ *(resp.* $\frac{n_0\,\tau_0}{\delta}$*). During the first (resp. second) phase, the revenue ratio of the miner is* $\frac{q'}{\delta}\cdot\frac{b}{\tau_0}$ *(resp.* $q\,\delta\cdot\frac{b}{\tau_0}$*) where* q' *and* δ *are given in Corollaries 2 and 3 (denoting by* δ *its expected value). Moreover, denoting by* R *(resp.* T*) the revenue (resp. duration time) at the end of an attack cycle, we have:*

$$\mathbb{E}[T] = \left(\delta + \frac{1}{\delta}\right) n_0\tau_0$$
$$\mathbb{E}[R] = (q' + q)n_0 b$$

Proof. The duration time and the revenue ratio of the attacker during the selfish mining phase has been computed in the previous section. After a first difficulty adjustment, the mining difficulty is divided by δ. So, blocks arrive on an average period $\frac{\tau_0}{\delta}$ and the duration time of the second phase is $\frac{n_0\,\tau_0}{\delta}$ (the miner mines honestly during this phase). Each time, there is a probability q that the attacker is the first one to discover the next block. Hence, during the second phase, his revenue ratio is $\frac{q\,b}{\frac{\tau_0}{\delta}} = \frac{q\,\delta b}{\tau_0}$. Finally, the revenue of the ISM miner at the end of an attack cyle is

$$\mathbb{E}[R] = \left(\frac{q'}{\delta}\,\frac{b}{\tau_0}\right)\cdot n_0\,\tau_0\,\delta + \left(\frac{q\,\delta\,b}{\tau_0}\right)\cdot\frac{n_0\tau_0}{\delta}$$

and the result follows.

Using Corollary 3, we then get the long-term apparent hashrate of the strategy.

Corollary 4. *The apparent hashrate of the Intermittent selfish strategy is* $q" = \frac{q+q'}{\delta+\frac{1}{\delta}}$*, and we have the closed form formula*

$$q" = \frac{q\left(1-4q^2+2q^3\right)\left(1+\gamma+(3-4\gamma)q+(5\gamma-11)q^2+(5-2\gamma)q^3\right)}{2-2q-11q^2+10q^3+18q^4-20q^5+5q^6}$$

Example 5. This formula agrees with simulations from [26]. The threshold ISM/HM is obtained when $q" = q$. When $\gamma = 0$, we get $q" = 0.365078$. The authors from [26] obtain 0.37 in their simulations.

Corollary 5. *When ISM is more profitable than honest mining, then SM is more profitable than ISM.*

Proof. By hypothesis, we have $\frac{q+q'}{\delta+\frac{1}{\delta}} > q$, which implies

$$\frac{q'}{q} > \delta + \frac{1}{\delta} - 1 > 1 > \frac{1}{\delta+\frac{1}{\delta}-1}$$

since $\delta + \frac{1}{\delta} > 2$ for $\delta > 0$. So, $\frac{q+q'}{\delta+\frac{1}{\delta}} < q'$ and ISM is less profitable than SM.

In Fig. 2 regions in $(q,\gamma) \in [0,0.5]\times[0,1]$ are colored according to which strategy is more profitable (HM is the honest mining strategy).

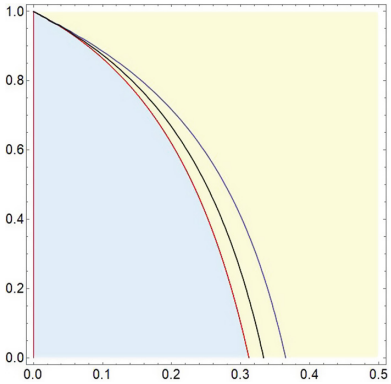


Fig. 2. Dominance regions in parameter space (q, γ). The threshold SM/HM (resp. ISM/HM, resp. SM/ISM) in black (resp. blue, red). When ISM is more profitable than HM then SM is always more profitable than ISM.

4.1 Profit Lag

As before, we denote by Δ the difference between the average revenue of selfish and honest mining. According to Proposition 5, after n attack cycles i.e., alternatively n phases of selfish mining and n phases of honest mining, the attacker earns $q" \cdot \left(\delta + \frac{1}{\delta}\right) n \cdot n_0\, b$ during $\left(\delta + \frac{1}{\delta}\right) n \cdot n_0 \tau_0$. So, at this date, we have

$$\Delta = (q" - q) \cdot \left(\delta + \frac{1}{\delta}\right) \cdot n\, n_0\, b$$

After this date, if the attacker selfish mines again, then at the end of this phase of duration $\delta n_0 \tau_0$, his revenue is $q" \cdot \left(\delta + \frac{1}{\delta}\right) n \cdot n_0\, b + \frac{q'}{\delta} \cdot n_0\, \delta\, \tau_0$. At this date, we have

$$\begin{aligned} \Delta &= (q" - q) \cdot \left(\delta + \frac{1}{\delta}\right) \cdot n\, n_0\, b + \left(\frac{q'}{\delta} - q\right) \cdot n_0\, \delta\, b \\ &= (q" - q) \cdot \left(\delta + \frac{1}{\delta}\right) \cdot n\, n_0\, b - (q\delta - q') \cdot n_0\, b \end{aligned}$$

and this quantity can possibly be negative depending on n. By the end of an attack cycle, we have on average $\Delta = (q" - q)\, n_0 \left(\delta + \frac{1}{\delta}\right) b$ which is positive when ISM is more profitable than honest mining. Thus when $q" > q$, ISM is more profitable than HM before a second difficulty adjustment as noticed in [26]. This comes as no surprise since the duration time of an attack cycle of

the ISM strategy corresponds to a period of 2×2016 official blocks, and any performant mining strategy is always profitable at the end of an attack cycle. See the discussion just after Definition 2.

However, after another difficulty adjustment the revenue falls because the difficulty increases and ISM becomes less profitable than HM. It is only after several difficulty adjustment periods of 2016 blocks that ISM can become definitely more profitable than HM. For instance, when $q = 0.1$ and $\gamma = 0.9$ this only happens approximately after the 13-th difficulty adjustment. Therefore, on average, it takes more than 6 months for the strategy to be definitely profitable, and this is much longer than for classical SM.

It is imperative to understand the unfolding of events. First, the selfish miner invests in lowering the difficulty, and, at any moment, he can get reap immediate profits, even before the profit lag, by just mining honestly. But if the miner wants to repeat the attack cycle, he will need to burn again these profits for the purpose of lowering the difficulty. This is what happens in the first cycles of the ISM strategy. In Fig. 3 we have the plot of the progression of Δ.

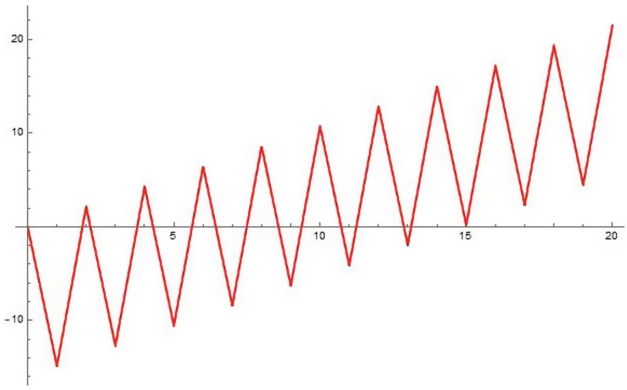


Fig. 3. Graph of $t \mapsto \mathbb{E}[R(t)] - qbt/\tau_0$ i.e., difference of average revenue between intermittent selfish and honest mining for $q = 0.1$ and $\gamma = 0.9$. X-axis: progression of the official blockchain, in difficulty adjustment units. Y-axis: revenue of the miner in coinbase units. The strategy is not definitely profitable before 13 difficulty adjustments. By comparison, SM is definitely profitable after 5 difficulty adjustments (compare with Fig. 1).

5 Alternate Network Mining Strategy

The strategy is described for alternate mining between BTC and BCH networks, but obviously it applies to any pair of networks with the same mining algorithm (with frictionless switching mining operations). This time we consider three distinct types of miners: the first ones mine on Bitcoin only, the second

ones on Bcash (BCH), and the third ones alternate between Bitcoin and Bcash. The ones mining alternatively between BTC and BCH are labelled "attackers" (although his strategy is legit and respects both network protocols). We assume that the revenue ratios of mining honestly on Bitcoin and Bcash are the same. This is approximately what is observed in practice since any divergence justifies a migration of hashrate from one network to the other. We denote by ρ this common value for the attacker. We assume also that there are no other miners and all miners mine with full power. In particular, the total hashrate remains constant. The attacker starts mining on Bitcoin at the beginning of a difficulty adjustment on BTC. An attack cycle is made of two phases. During Phase 1, the attacker withdraws from BTC and mines on BCH until n_0 blocks have been mined on Bitcoin. During Phase 2, the attacker comes back to mine on the BTC network until a new difficulty adjustment. We call this strategy: *Alternate Mining Strategy*. This is a variation of smart mining. The only difference being that the miner does not remain idle during Phase 1 but goes on to mine on BCH. Note that this mining strategy does not provoke periodic reorganizations of the blockchain. The main annoyance for users of both networks is that difficulty does not stabilize and for Bitcoins users blocks arrive regularly at a slower pace than in the steady regime with the miner fully dedicated to the BTC network.

We denote by δ the difficulty adjustment parameter after Phase 1. Since the miner comes back on BTC in the second phase, the second difficulty adjustment at the end of the second phase is $\frac{1}{\delta}$.

Proposition 6. *The duration phase of Phase 1 (resp. Phase 2), is $n_0\tau_0\delta$ (resp. $n_0\tau_0\frac{1}{\delta}$). During Phase 1 (resp. Phase 2), the revenue ratio of the attacker is ρ (resp. $\rho\delta$). The revenue ratio of the alternate mining strategy is $\frac{1+\delta}{\delta+\frac{1}{\delta}}\rho$.*

Proof. By definition of the difficulty adjustment parameter, the duration time of Phase 1 (resp. Phase 2) is $\delta n_0\,\tau_0$ (resp. $\frac{1}{\delta}n_0\,\tau_0$). So the duration time of an attack cycle is $\left(\delta+\frac{1}{\delta}\right)\,n_0\tau_0$. During Phase 1, the attacker's revenue ratio is ρ because we assume that the attacker mines honestly during this phase with the assumption that the revenue ratio is the same for BCH and BTC. During Phase 2, the mining difficulty is divided by δ. So, the revenue ratio of the attacker during this period is $\delta\cdot\rho$. Therefore, the revenue of the attacker after an attack cycle is $\rho\cdot n_0\,\tau_0\,\delta+(\rho\,\delta)\cdot\frac{n_0\tau_0}{\delta}$.

Corollary 6. *The alternate mining strategy is always more profitable than honest mining and selfish mining for all values of q.*

Proof. The first statement results from $\delta>1$ that implies $\frac{1+\delta}{\delta+\frac{1}{\delta}}>1$. To prove, the second statement, we remark that in Phase 1, blocks are only validated by honest miners. So this phase lasts on average $\frac{n_0\tau_0}{p}$ and the difficulty parameter is updated accordingly: $\delta=\frac{1}{p}$ with $p=1-q$. So, the revenue ratio of the attacker is $\frac{2-q}{2-2q+q^2}\frac{b}{\tau_0}$ as follows by replacing δ with $\frac{1}{p}$ in the formula from Proposition 6. On the other hand, in the most favorable case (when $\gamma=1$), the revenue ratio

of the selfish miner is $\frac{q\left(2q^3-4q^2+1\right)}{q^3-2q^2-q+1}\frac{b}{\tau_0}$. Then, the result comes from $\frac{2-q}{2-2q+q^2} > \frac{q\left(2q^3-4q^2+1\right)}{q^3-2q^2-q+1}$ for $0 < q < 0.5$ that we prove studying the polynomial

$$(2-q)\cdot(q^3-2q^2-q+1)-(2-2q+q^2)\cdot\left(q\left(2q^3-4q^2+1\right)\right)$$

This polynomial is non-increasing on $[0, \frac{1}{2}]$ and remains positive on this interval.

5.1 Profit Lag

As before, we denote by Δ the difference of the average revenue between selfish mining and mining honestly from the beginning.

Proposition 7. *After n attack cycles, we have $\Delta = \rho\cdot\left(1-\frac{1}{\delta}\right) n\, n_0\tau_0 = q^2\, n\, n_0 b$ and Δ stays constant during the BCH mining phase.*

Proof. After a first phase of mining, we have $\Delta = 0$ since the revenue ratio on BCH and BTC are assumed to be equal. At the end of the second phase, we have $\Delta = \left(1-\frac{1}{\delta}\right)\rho\cdot n_0\tau_0$. Indeed, the second phase lasts $\frac{n_0\tau_0}{\delta}$ and the revenue ratio of the attacker during this phase is $\rho\delta$. Now, we use that $\rho = q\frac{b}{\tau_0}$ and $\delta = \frac{1}{p}$ because only honest miners are mining during Phase 1. The strategy is then a repetition of altering phases 1 and 2 and the result follows.

We plot the graph of Δ in Fig. 4.

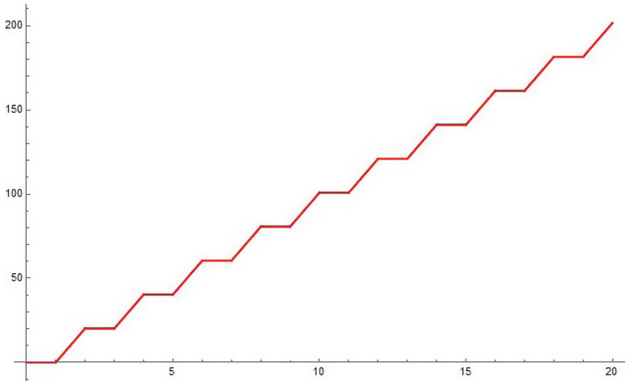


Fig. 4. Difference of average revenue between alternate and honest mining for $q = 0.1$. X-axis: progression of the official blockchain, in difficulty adjustment units. Y-axis: revenue of the miner in coinbase units. The strategy is definitely profitable after one difficulty adjustment.

6 Conclusion

We revisit the selfish and intermittent mining strategies, and define alternate mining strategy for pairs of networks with the same PoW. We clarify the profitability setup using the new notion of "profit lag". We compute exact formulas for different strategies of the profit lag and the revenue ratio. We show that, under natural hypothesis, the alternate mining strategy is the best one: its revenue ratio is the largest and the profit lag is the least. So, the existence of two networks sharing the same PoW algorithm can lead to instabilities of the difficulty and with blocks validated slower than normal.

References

1. Antonopoulos, A.: Mastering Bitcoin: Unlocking Digital Cryptocurrencies. O'Reilly Media Inc (2014)
2. Bahack, L.: Theoretical bitcoin attacks with less than half of the computational power (draft) (2023). arXiv:1312.7013
3. Bahack, L., Courtois, N.: On Subversive Miner Strategies and Block Withholding Attack in Bitcoin Digital Currency (2014). arXiv:1402.1718
4. Buterin, V.: Change difficulty adjustment to target mean block time including uncles (2017). https://github.com/ethereum/EIPs/issues/100
5. Douc, R., Moulines, E., Priouret, P., Soulier, P.: Markov Chains. Springer, Berlin (2020)
6. Dryja, T., Poon, J.: The bitcoin Lightning Network: scalable off-chain instant payment (2016). https://lightning.network/docs/
7. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: 12th Annual International Cryptology Conference (1992)
8. Eyal, I., Sirer, E.: Majority is not enough: bitcoin mining is vulnerable. In: International Conference on Financial Cryptography and Data Security (2014)
9. Felten, E.: Bitcoin isn't so broken after all (2013). https://freedom-to-tinker.com/2013/11/07/bitcoin-isnt-so-broken-after-all
10. Feng, C., Niu, J.: Selfish mining in ethereum. In: IEEE 39th International Conference on Distributed Computing Systems (2019)
11. Georgiadis, G., Zeilberger, D.: A combinatorial-probabilistic analysis of bitcoin attacks. J. Differ. Equ. Appl. (2019)
12. Goren, C., Spiegelman, A.: Mind the mining. In: Proceedings of the 2019 ACM Conference on Economics and Computation (2019)
13. Grunspan, C., Pérez-Marco, R.: Double spend races. Int. J. Theor. Appl. Financ. **21** (2018)
14. Grunspan, C., Pérez-Marco, R.: On profitability of selfish mining (2018). ArXiv:1805.08281v2
15. Grunspan, C., Pérez-Marco, R.: Bitcoin Selfish Mining and Dyck Words (2019). ArXiv:1902.01513
16. Grunspan, C., Pérez-Marco, R.: Selfish Mining and Dyck Words in Bitcoin and Ethereum Networks. Tokenomics, International Conference on Blockchain Economics, Security and Protocols, Paris (2019)
17. Grunspan, C., Pérez-Marco, R.: On profitability of nakamoto double spend. Probab. Eng. Inf. Sci. (2021)

18. Grunspan, C., Pérez-Marco, R.: The mathematics of bitcoin. Europ. Math. Soc. Newslett. (2020)
19. Grunspan, C., Pérez-Marco, R.: Selfish mining in ethereum. In: International Conference on Mathematical Research for Blockchain Economy. Imperial College London (2020)
20. Ily, D., Werner, S., Knottenbelt, J.: Unstable throughput: when the difficulty algorithm breaks. In: IEEE International Conference on Blockchain and Cryptocurrency (ICBC) (2021)
21. Khosravi, M.: A full-fledged simulation for selfish mining in bitcoin. https://armankhosravi.github.io/dirtypool2/
22. Nakamoto, S.: Bitcoin P2P e-cash paper. https://www.metzdowd.com/pipermail/cryptography/2008-October/014810.html
23. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). https://bitcoin.org/bitcoin.pdf
24. Nakamoto, S.: Bitcoin v0.1 released. https://www.metzdowd.com/pipermail/cryptography/2009-January/014994.html
25. Nayak, K., Shi, E., Kumar, S., Miller, A.: Stubborn mining: generalizing selfish mining and combining with an eclipse attack. IEEE European Symposium on Security and Privacy, pp. 305–320 (2016)
26. Negy, K., Rizun, P., Sirer, E.: Selfish mining re-examined. In: International Conference on Financial Cryptography and Data Security (2020)
27. RHorning: Mining cartel attack (2010). https://bitcointalk.org/index.php?topic=2227.0
28. Rosenfeld, M.: Analysis of Bitcoin Pooled Mining Reward Systems (2011). arXiv:1112.4980
29. Rosenfeld, M.: Analysis of hashrate-based double spending (2014). arXiv:1402.2009v1 2018
30. Song, J.: Programming Bitcoin: Learn How to Program Bitcoin from Scratch. O'Reilly Media (2019)
31. Wattenhofer, R.: Blockchain Science: Distributed Ledger Technology. Independently Published (2019)
32. Wuille, P.: Assuming the Bitcoin hashrate is perfectly constant, and all blocks have exact timestamps (corresponding to the time they were mined). Which of the options below is closest to the expected time retargetting periods will take? (2019). https://twitter.com/pwuille/status/1098288749098795008



Oracle Counterpoint: Relationships Between On-Chain and Off-Chain Market Data

Zhimeng Yang[1], Ariah Klages-Mundt[2(✉)], and Lewis Gudgeon[3]

[1] Coinbase, London, UK
[2] Cornell University, Ithaca, NY, USA
aak228@cornell.edu
[3] Imperial College London, London, UK

Abstract. We investigate the theoretical and empirical relationships between activity in on-chain markets and pricing in off-chain cryptocurrency markets (e.g., ETH/USD prices). The motivation is to develop methods for proxying off-chain market data using data and computation that is in principle verifiable on-chain and could provide an alternative approach to blockchain price oracles. We explore relationships in PoW mining, PoS validation, block space markets, network decentralization, usage and monetary velocity, and on-chain Automated Market Makers (AMMs). We select key features from these markets, which we analyze through graphical models, mutual information, and ensemble machine learning models to explore the degree to which off-chain pricing information can be recovered entirely on-chain. We find that a large amount of pricing information is contained in on-chain data, but that it is generally hard to recover precise prices except on short time scales of retraining the model. We discuss how even noisy information recovered from on-chain data could help to detect anomalies in oracle-reported prices on-chain.

Keywords: Oracles · DeFi · On-chain data · Blockchain economics · Ensemble learning

1 Introduction

Decentralized finance (DeFi) aims to transfer the role of trusted but risky intermediaries to more robust decentralized structures. A remaining weak link is in reliance on off-chain information, such as prices of reference assets, which need to be imported on-chain through oracles. The issue is that oracle-reported prices cannot be proven on-chain because the price process (usually in USD terms) is not observable there.

Various oracle security models exist, as described in [17], though for the most part, they always involve some sort of trusted party or medianizing of several trusted parties. Even alternatives like referencing time weighted average prices (TWAPs) on decentralized exchanges (DEXs) still essentially involve a trusted

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
P. Pardalos et al. (Eds.): MARBLE 2023, 2024.
https://doi.org/10.1007/978-3-031-48731-6_8

party. In particular, to price an asset in USD terms, the standard approach is to use a DEX pair with a USD stablecoin, which shifts the trusted party to the stablecoin issuer as a sort of oracle. While real trades can be observed on-chain, which is a strength of this method, stablecoin issuers can become insolvent, censor transactions, or otherwise cause the stablecoin to depeg. A resilient oracle system would ideally handle the corner cases of stablecoin pricing.

In this paper, we explore a new direction in oracle design wherein an *estimate* of an off-chain price can in principle verifiable on-chain, which is intended to be used in addition to other oracle methods. We investigate the theoretical and empirical relationships between activity in on-chain markets and the overall pricing and liquidity in off-chain cryptocurrency markets (e.g., {BTC, ETH}/USD price. The motivation is to develop methods for proxying off-chain market data using data within an on-chain environment.

We formalize this as the task of finding a function f that maps on-chain observable data (or features for the machine learning model) to close estimates of off-chain prices, as visualized in Fig. 1a. We consider a variety of features: *basic features* that are straightforward properties about the network, *economic features* that are computed from basic features and are suggested by fundamental economic models of the network, and features that arise from the usage of DeFi protocols. Ideally, a good f will also have two further properties: (i) it is difficult/costly to manipulate the output of f through manipulating the inputs, and (ii) outputs of f are provable on-chain. The hypothesis predicating this structure is that off-chain price data (e.g., in USD terms) is incorporated into the behavior of agents in on-chain markets (e.g., mining, block space, and DeFi markets) and that on-chain data thus provides some information that can be recovered about the original off-chain prices, as visualized in Fig. 1b.

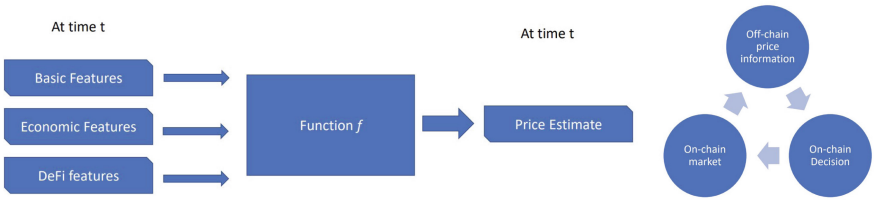


Fig. 1. Proposed structure to estimate prices verifiably on-chain.

To understand the problem intuitively, compare with the usual financial price prediction problem, in which we would try to identify several drivers of future price and formulate a model to predict future prices with these drivers as features. The problem we consider is the reverse in some ways. In particular, we hypothesize that the price is a driving factor (probably one of many) behind the behavior of agents in on-chain markets, and we want to recover the current period price from the current state of on-chain market behaviors as features.

We explore this problem using a combination of economic theory about on-chain markets and data-driven analysis to explore the degree to which off-chain pricing information can be recovered from on-chain data. We find a meaningful price signal is recoverable as well as several strong empirical relationships with on-chain features. While it is not precise enough to use directly as an oracle, we discuss ways in which it could be used as a trustless sense check for oracle-reported prices. We finish by discussing several significant challenges that remain in developing and executing such a tool.

2 Methods

We explored relationships between off-chain ETH/USD and CELO/USD[1] prices and feature variables from PoW mining for Ethereum and Bitcoin (alternatively PoS validation for Celo), block space markets, network decentralization (e.g., burden on running a full node), usage and monetary velocity, and DeFi liquidity pools and AMMs, including activity on Bitcoin, Ethereum, and Celo networks. We obtained raw block and transaction data from Google Cloud Bigquery, Uniswap v1 and v2 data from the Graph, and off-chain USD price data from the Coinbase Pro API. We then derived the following types of on-chain data features:

- *Basic network features* that can be straightforwardly derived from Ethereum block and transaction data, covering information related to Ethereum's network utility, ether supply, transaction cost and the network's computational consumption (i.e. the gas market).
- *Uniswap features* on participation in DEX pools involving ETH and stablecoins (DAI, USDC, USDT). For the most part, we intentionally do not focus on DEX prices, as those measures would equivalently treat the stablecoin issuer as a sort of trusted oracle. We instead mainly focus on a measure of liquidity moving in and out of DEX pools.
- *Economic features* that are suggested by fundamental economic models of decentralized networks and can also be derived from on-chain data (as described in the next subsection).

Data was collected spanning from July 1 2016 to May 1 2022 and was aggregated to the hourly level. We include Bitcoin data along with Ethereum data in the dataset for the sake of exploring relationships as in principle it can also be verified on-chain to varying degrees and discuss the connections further later.

Some further details on data and features are provided in the appendix. Precise methods are available in the project github repo: https://github.com/tamamatammy/oracle-counterpoint.

[1] ETH/USD is the focus as it is the main oracle input for DeFi protocols and is explored for PoW Ethereum data. CELO/USD is included as a first look at PoS data.

2.1 Fundamental Economic Features from On-Chain Markets

In addition to the above raw on-chain features, we also considered transformations of these features informed by fundamental economic models of on-chain markets, including PoW mining, PoS validation, block space markets, network decentralization costs of running full nodes, usage and monetary velocity, and on-chain liquidity pools (e.g., [2–4,6,8,16]). We analyzed the structure of these models to extract features that should economically be connected to price.

For example, [8] models a block space market and finds that the ratio of average demand to capacity $\rho = \frac{\lambda}{\mu K}$ plays an important role in linking users' waiting costs to transaction fees pricing. Here λ is the transaction volume, K is the maximum number of transactions in a block, and μ is the block adding rate. A function emerges, called $F(\rho)$ that describes the relationship between fee pricing and congestion (i.e., amount of demand compared to supply for block space), which can be translated as

$$\text{tx fees in USD} = (\text{tx fees in ETH}) * \text{price}_{ETH} = F(\rho).$$

While $F(\rho)$ is nontrivial to work with, various pieces of the results in [8] can be incorporated into useful features for the task of recovering price_{ETH} (i.e., ETH/USD), including ρ, ρ^2, and the empirical finding that $\rho = 0.8$ represents a phase transition in fee market pricing.

We also used the model in [2], which modeled cryptocurrency price based on market fundamentals. A key feature in their model was currency velocity, which is defined as the ratio of transaction volume to money supply:

$$\text{Velocity} = \frac{\text{Transaction Volume (USD)}}{\text{Exchange Rate} * \text{Supply of Currency}}.$$

In their model, they show a unique equilibrium cryptocurrency exchange rate based on supply and demand with steady state expected exchange rate equal to the ratio of expected transaction volume and cryptocurrency supply. Based on this model, we also incorporated the ratio of transaction volume (in cryptocurrency units since USD value is not given) and cryptocurrency supply, measured over one hour time steps, as an additional feature variable in our analysis.

We formulate other factors related to mining payoff, computational burden, and congestion as reviewed in the appendix.

2.2 Data-Driven Feature Analysis

We analyse empirical relationships between features using graphical models and mutual information to study which features are most related to USD prices.

We use probabilistic models of Markov random fields, generated through sparse inverse covariance estimation with graphical lasso regularisation over normalized data, to express the partial/pair-wise relations between the time series of on-chain feature variables and off-chain prices. For example, if the true underlying structure is Gaussian, then entries of the inverse covariance matrix are zero if

and only if variables are conditionally independent (if the underlying structure is not Gaussian, then in general the inverse covariance matrix gives partial correlations).[2] Sparse inverse covariance estimation is a method for learning the inverse covariance structure from limited data by applying a form of L1 regularization (see [7]).

The output of this technique helps to uncover strong empirical dependencies within the data, suggesting features that are strongly related to price and others that replicate similar information as others. We find that the method is often sensitive to the precise dataset used, which we adjust for by smoothing over the outputs of many k-fold subsets.

We also consider mutual information between features in the dataset, which describes the amount of information measured (in information entropy terms), measured in reduction of uncertainty, obtained about price by observing the on-chain features. In information theory, entropy measures how surprising the typical outcome of a variable is, and hence the 'information value'. This is helpful both in identifying strong relationships and evaluating different smoothing factors considering noisy on-chain signals. In this analysis, we consider smoothed versions of the feature set based on exponential moving averages with memory parameters α, i.e., for feature value b_t at time t, the smoothed measure is

$$\tilde{b}_t = (1 - \alpha)b_t + \alpha\tilde{b}_{t-1}.$$

2.3 Modeling Off-Chain Prices

We apply supervised machine learning methods to explore the degree to which off-chain pricing information can be recovered from information that is entirely on-chain. We apply a few select simple and ensemble supervised machine learning methods on a rolling basis: basic regression, single decision tree, random forest, and gradient boost. A decision tree is a non-parametric method that breaks down the regression into subcases of the feature variables following a tree structure. A tree ensemble method constructs many tree models and uses the average of the trees as the output. The motivation for using tree-based ensemble methods is the non-parametric nature of the dataset and success of similar methods in analyzing other market microstructure settings [5]. In our results, we focus on random forest, which trains many trees to different samples of the empirical distribution of training points, as this method tends to be resilient to overfitting.

We run these models on the data set and evaluate performance using out-of-sample testing data on a rolling basis. The rolling training-testing data split, as depicted in Fig. 2, is applied to boost model performance. For a given set of time series data with time duration of time t + time c = time t+c, where time series before time t were used for model training and time series between time t and time t + c were used for model testing. The benefit of this split is to test how good the model is in proxying ETH USD price for a fix period in the future, with all the information available in the past.

[2] In comparison, covariance represents relations between all variables as opposed to partial/pairwise relations.

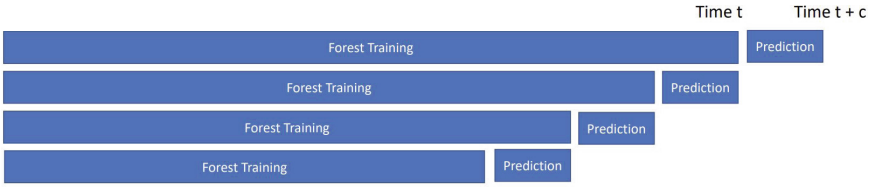


Fig. 2. Rolling training-testing data split

3 Results

We focus on Ethereum data analysis under PoW in this section. Analysis of Celo data is included in the appendix as a first look at a PoS system. There is not yet enough historical data to analyze Ethereum PoS but would be a next step.

3.1 Feature Analysis

We find that a large amount of off-chain pricing information is contained in on-chain data and that the various features are connected in some strong but complicated ways.

Figure 3 and Appendix Fig. 7 show the results of sparse inverse covariance modeling for a selection of the feature set. The graphical structure depicted is the consistent structure over time as smoothed over the outputs of many k-fold subsets. The partial correlation matrix shows the graphical structure in matrix form. The features that are most directly connected with ETH/USD price, as measured by partial correlations in the graphical model, include number of active to and from addresses sending transactions, block difficulty, and number of transactions per block. Several of these variables appear to contain information relevant to price as well, as measured by mutual information in the next analysis. The graphical model suggests that these are indirect relationships, though it is worth noting that the process is unlikely Gaussian, and so the partial correlations do not necessarily translate to conditional dependencies.

Figure 4 shows the mutual information between ETH/USD prices and other features, meaning the amount of information (reduction of uncertainty) obtained about price by observing each other variable individually. We find that across the top 10 features, a large amount of information about off-chain price is contained in on-chain data. We also find that the mutual information decreases with α, the exponential moving average memory factor for smoothing, indicating that the smoothed data is generally less informative than the most up-to-date data.

We also analyze the full feature set, including the transformed economic factors and Uniswap pool liquidity factors. Perhaps unsurprisingly, since the transformed features contain the same underlying information, they do not exhibit stronger relationships than the raw features. More surprising is that the Uniswap pool factors also did not present strong relationships with price. We then arrived

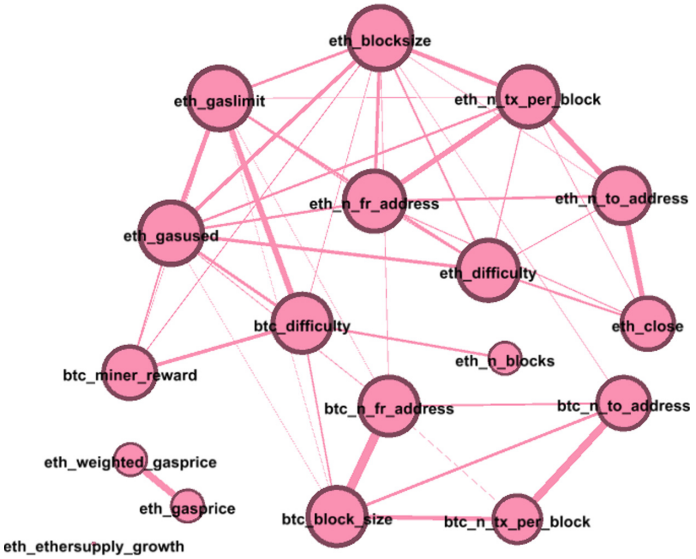


Fig. 3. Graphical network visualization. Variables defined in Table 4.

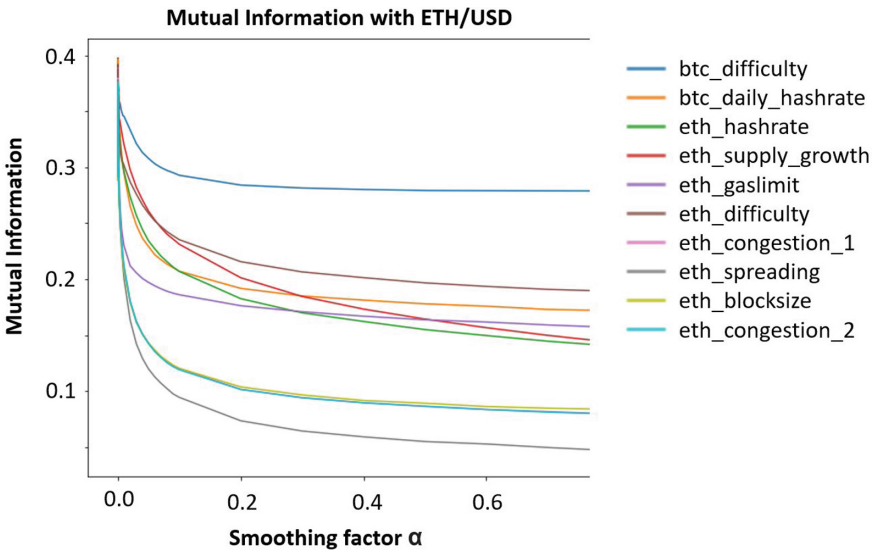


Fig. 4. Mutual information of price data and feature variables, with memory parameter (or smoothing factor) α applied to the feature variables (see Sect. 2.2). Variables defined in Table 4.

at the above version of the analysis excluding Uniswap factors enabling us to use the entire data history (as Uniswap was launched later than the start of the dataset).

3.2 Recovering Off-Chain Prices from On-Chain Data

Random forest and gradient boost both outperformed the other two simpler ML algorithms. We selected Random Forest as the candidate model in the end as it is in principle simpler to be implemented on-chain compared to the gradient boost model (theoretically, a random forest model could be implemented as one big mapping table in a smart contract).

We tested the model performance over different lengths of period—the length of time duration between time t and time t+c. As would be expected with nonstationary time series, we observed that the longer the time duration that a single trained model is used for price estimation, the less accurate is price estimation. The degree to which time between retrainings affects accuracy is informative, however.

Figure 5 shows the random forest model performances, Estimated vs Actual ETH/USD price, for 1-day ahead, 1-week ahead and 1-month ahead of retrainings. While none of the models provide high accuracy of recovering ETH prices, they do demonstrate that a good signal of the general price level can be recovered, particularly in the 1-day and somewhat in the 1-week retraining cases.

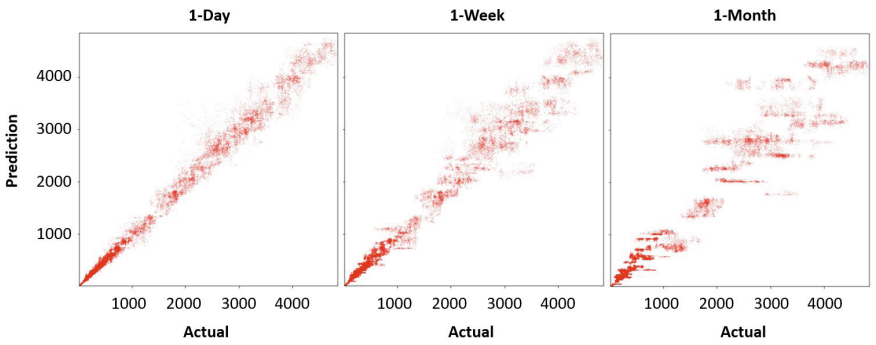


Fig. 5. Recovered price vs actual for random forest with given retraining periods.

The deviation between estimated price and actual price is bigger for higher ETH prices. This is a combination of both having less data in the dataset for these prices and the fact that the same relative error scales with the absolute price, and so deviations measured absolutely are expected to be greater.

We run the models on the full feature set, including transformed economic factors and Uniswap pool factors. The economic factors provide little new information vs the raw features, perhaps a consequence of the flexibility of the tree models. Uniswap pool factors similarly do not improve accuracy. The final analysis excludes Uniswap factors enabling the entire data history to be used.

3.3 Performance of Price Recovery

To measure performance of the price recovery models, we compare against a simple martingale benchmark. This benchmark supposes that the last observed price in the last retraining period is the best estimate of the next price in expectation, barring any new information, which would follow in an efficient market. By comparing against this benchmark, we evaluate how well the on-chain feature variables, the only source of new information to the model, recover price vs the best guess without this information.

We evaluate the squared error between a prediction (either the model or benchmark price) at time t and the actual price as

$$SE = (\text{predicted}/\text{actual} - 1)^2.$$

We then consider the mean squared error (MSE) over different times t. The square root of the MSE (RMSE) then gives a measure of error that can be interpreted as a percentage of the price level.

We compare model errors with benchmark errors using these measures. We first consider the difference in squared errors between the model and the benchmark as

$$DSE = (\text{benchmark}/\text{actual} - 1)^2 - (\text{model}/\text{actual} - 1)^2.$$

This quantity will be positive when the model performs better than the benchmark and negative otherwise. Table 1 summarizes how frequently the models have lower squared error than the benchmarks and by how much the squared error is reduced (as a percentage of price level) when this happens. Note that most of the time, the models perform worse than the benchmarks over the dataset. However, they may be able to provide useful information in addition to the benchmarks in some settings.

Table 1. Summary of DSEs between models and benchmarks for different retraining periods evaluated on the whole dataset (2016–2022). Row 1 is the frequency that $DSE > 0$. Row 2 is the root mean DSE at the times that $DSE > 0$.

Model retraining periods	1-day (%)	7-day (%)	30-day (%)
How often model beats benchmark	12.4	26.9	32.4
Gain over benchmark when model is better	0.65	3.56	7.10

In line with Table 1, the MSE of the models is greater than the MSE of the benchmarks when taken over the entire dataset. A limitation with this measure is that, for data points early in the dataset, there is relatively less training data for the rolling models. We might expect the models to do better toward the end of the dataset where there is more training data to work with.

In Table 2, we calculate RMSEs restricted to the last year of the dataset, when the models can theoretically be best. In addition to RMSE over this time period,

we also compute RMSE during the top 10% most volatile days, as measured by rolling 24 h volatility calculated on hourly returns.

Table 2 shows some limited situations where a model makes improvements over the benchmark as measured in less RMSE compared to the benchmark. This happens for the 30-day retraining model, which also tends to perform better for the most volatile days. In general, the model error is usually larger than the benchmark error, however, and the outperformance of the 30-day model is somewhat sensitive to the restriction to the final year, which is further explored in Appendix B.1.1.

Table 2. RMSEs of the models compared to benchmarks over the last year of the dataset (May 2021–May 2022).

Model retraining periods	1-day (%)	7-day (%)	30-day (%)
Model RMSE	7.82	18.83	18.98
Benchmark RMSE	3.77	9.39	19.80
Model (top 10% vol) RMSE	15.41	23.15	29.84
Benchmark (top 10% vol) RMSE	7.5	12.13	36.61

While the current models hint that on-chain information could be useful in reducing error, in practice they are not precise enough. In particular, an application could get most of the utility of the current models by checking that oracle prices don't change too quickly (i.e., implement a check of current oracle prices against the benchmark estimate of a previously observed price). Note that the last observed price in the last retraining period is not an input variable to the price recovery model other than as part of training. A next step in improving model performance could be to incorporate this last training price or to train the model to predict how current price deviates from the last training price. More refined methods could find ways of extracting better information from the on-chain data.

4 Discussion

We find that a general, but noisy, signal of off-chain prices can be extracted from the on-chain feature set, although it remains difficult to extract precise prices from the noise. It is possible to improve the accuracy of the model by including features of DEX pricing of ETH/stablecoin pairs, as would be expected from [1]. However, this would implicitly rely on the assumption that 1 stablecoin = 1 USD, which would face the significant further issues of detecting stablecoin depeg events (such as happened in USDC in March 2023) given that data is sparse for such events. Instead, the aim of this work is to provide information that can be used on top of existing oracle mechanisms, including DEX pricing, to relax trust requirements in those methods.

While this approach could likely not be used as a direct price oracle, the information from the recovered price signal could still in principle be useful as a sense check to inform when other oracle-reported prices may be suspect. This function would be potentially very useful in application as the most profitable oracle manipulations to date have been large manipulations that may be caught by such methods. An existing oracle system of this style in [13] has been developed by cross-referencing information from DEX price sources. This approach has limitations, however, in sense checking the connection to the desired quote asset USD since it cannot be represented by DEX prices alone. Incorporating measures of the price signal that we uncover on top of the existing structure could help to mitigate this limitation. A main open question here is whether measures of price signal reported on-chain can be improved enough to make this feasible in application as a means of anomaly detection for oracle-reported prices. Such a method could also serve to better align the incentives of an oracle provider to report correct prices with the knowledge that their quality of their feed is being graded against the signal in on-chain information. Models such as [9,12] could model this analytically, interchanging the oracle provider with the governors in those models.

Several challenges remain for implementing and running such a mechanism in practice. One is accessing all the data within the EVM. Some of the data is in principle possible to access but may be too computationally intense under current systems. For instance, proving information about transactions or bridging BTC data might require running light clients on-chain. For example, see [10,11] for possible methods of referencing normally difficult-to-obtain features of a chain from a smart contract. For BTC data, this can mostly be ignored as it was not critical for the predictive models, but there was a lot of information in Ethereum transaction statistics. It is worth noting that some features such as gas prices are easier to access now with EIP 1559. Another challenge is in evaluating how manipulable the features are should a bad actor want to affect the price estimation. In principle, resilient measures seem possible considering that on-chain markets can be costly to manipulate, though may also be computationally burdensome to produce.

An implementation would also have to handle the rolling nature of retrainings required to accurately recover price data. The implementation would need a trust minimized way to update a smart contract implementation with new trainings. In principle this is also possible, such as by implementing the training program in fixed point to run deterministically and implementing a way to prove the correctness of a training on-chain. However, this would be daunting from the technical side as well as likely costly to run in most environments. The burden could possibly be eased by running it 'optimistically' by incorporating a challenge period and fraud proofs, though it's unclear if this would be enough of an improvement. Another viable way is for a trusted trainer to regularly update calibrations on-chain subject to on-chain spot checks and not full proofs.

Acknowledgements. This paper is based on work supported by a Bloomberg Fellowship, EPSRC Standard Research Studentship (DTP) (EP/R513052/1) and a Celo grant.

A More Details on Dataset Features

Figure 6 and Table 3 provide more information on the feature set used.

	CCY	Source	Starting From	Frequency
On-Chain	BTC block data	Google BigQuery	Jan 2016	Hourly
	ETH block data			
	CELO block data	Celo Graph (block, celoTransfers)	Apr 2020	
	cGLD transaction data			
	cUSD transaction data			
	Uniswap liquidity and balance data	The Graph	Aug 2020	
Off-Chain	BTC price and volume data	Coinbase API	Jan 2016	
	ETH price and volume data			
	Celo price and volume data		Sep 2020	

Fig. 6. Overview of dataset.

Table 3 describes the full feature variable set used at a high level, including basic network features, DeFi features from Uniswap, and features informed from economic models.

Table 3. Data features (high level).

Feature type	Feature (high level description)
Network	Number of blocks
	Number of transactions
	% change in accumulated ETH supply
	Avg gas limit
	Avg gas used
	Avg gas price
	Hash rate
Uniswap	Liquidity in ETH/stablecoin pools
	Trade volume in ETH
Economic	Mining pay-off factors
	Computational burden measures
	Congestion factors
	Social cost factors
	Spreading factor

Table 4 defines the variables used in the main text figures.

Online documentation in the project github repo provide further details of the underlying economic models and calculation of the economic factors (as well as calculation of other factors from the raw data): https://github.com/tamamatammy/oracle-counterpoint.

Table 4. Definitions of variables used in figures.

Feature	Definition
`eth_gaslimit`	hourly average gas limit on Ethereum
`eth_gasused`	hourly average gas used per block
`eth_blocksize`	hourly average Ethereum blocksize (gas)
`eth_n_from_address`	hourly average sender addresses per block
`eth_n_to_address`	hourly average receiver addresses per block
`eth_supply_growth`	hourly change in ETH supply
`eth_hashrate`	hourly average hashrate on Ethereum
`eth_gasprice`	hourly average tx gas price
`eth_weighted_gasprice`	hourly avg tx gas price, weighted by gas used in tx/total gas used
`eth_congestion_1`	hourly gas used/hourly gas limit
`eth_congestion_2`	square of `eth_congestion_1`
`eth_spreading`	hourly # receiving addresses/hourly # sender addresses
`eth_close`	hourly ETH/USD closing price
`btc_difficulty`	hourly average difficulty on Bitcoin
`btc_minter_reward`	hourly average miner reward on Bitcoin
`btc_n_from_address`	hourly average sender addresses per block
`btc_block_size`	hourly average Bitcoin block size (bytes)
`btc_n_tx_per_block`	hourly average # txs per block
`btc_to_address`	hourly average receiver addresses per block
`btc_daily_hashrate`	hourly Bitcoin difficulty/hourly average block time

A.1.1 Economic Features

A brief overview of the features informed by fundamental economic models is as follows along with citations for the relevant models that influenced the choice of these features.

- Mining payoff factor 1: $(R(\text{blockReward} + \text{blockFees}))^{-1}$ [14,16]
  - R = block rate (/s), eth_n_blocks = # blocks in the last hour
- Previous high hash rate/current hash rate
- previous high $(R(\text{blockReward} + \text{blockFees}))^{-1}$/current
- Excess block space (block limit—gas used)
- Social value: D(W) is the social value of the level of decentralization = D(W) = − log(W) $\implies$ D(W) = − log(gas_used) for ethereum, = − log(bytes); gas used as the measure of the weight of a block (W) [3]
- Social cost: Marginal cost = 1/gas_used or 1/bytes [3]
- Computational burden on nodes: use block_size as bandwidth $\implies$ block_size $*$ $\log^2$(block_size) [3]
- Congestion factors: rho = gas used/gas limit, and rho^2; (in economic model, rho is defined as average number of transaction per block/number of transactions per block) [8]

- Congestion factor: Indicator_{rho $> x$}, heuristic use x = 0.8 [8]
- Congestion pricing term 1: F(rho)/tx_fees_eth, where F describes relationship between USD tx fees and congestion [8]
  - Heuristic: use F = congestion factor 1 or 2 above
- Congestion pricing term 2: max number of transactions in a block/fees in block [15]
- Congestion pricing term 3: max number of transactions squared in a block/fees in block [15]
- Spreading factor: number of unique output addresses/number of unique input addresses [2].

B Further Information on Ethereum Analysis

Sparse inverse covariance estimation was performed with the implementation in SciPy using an alpha parameter of 4, convergence tolerance of 5e-4, 5 folds for cross validation, 4 grid refinements, and 1000 max iterations.

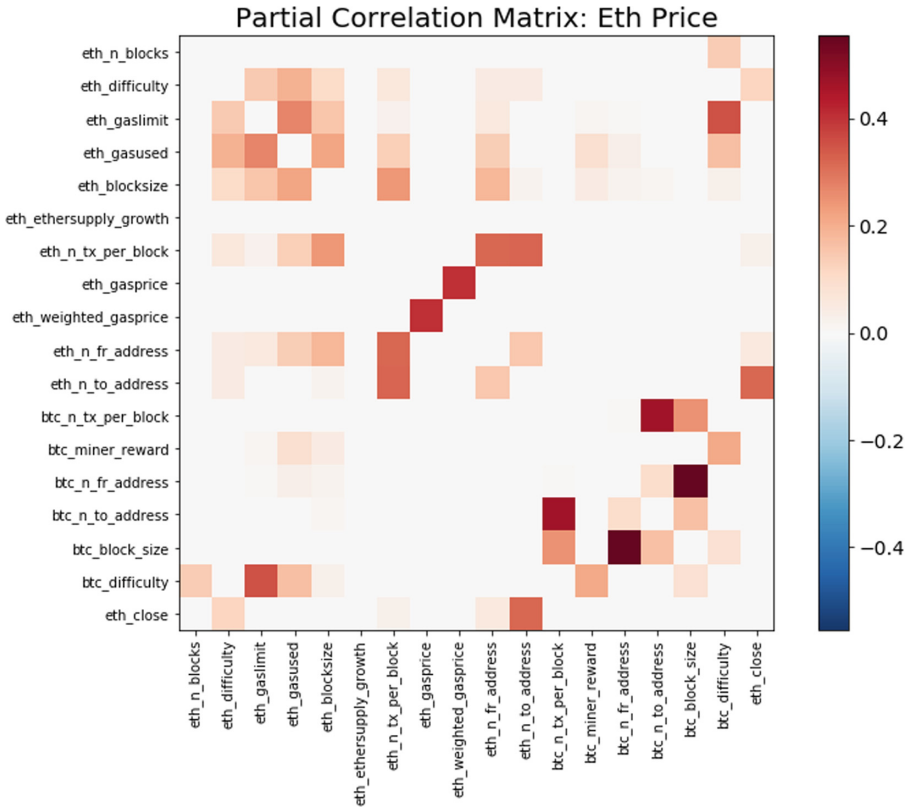


Fig. 7. Partial correlation matrix from sparse inverse covariance estimation.

B.1.1 Performance of Price Recovery

We continue the analysis from Sect. 3.3 with a few additional charts that give a wider view on performance for the 30-day rolling retraining model. Figure 8 shows the difference in squared error of the benchmark minus the model over time. When $DSE > 0$, the model is performing better than the martingale benchmark of the last observed price in the rolling retraining. Of note is that while the performance improves in the final year (May 2021–May 2022), one small period represents most of the performance size.

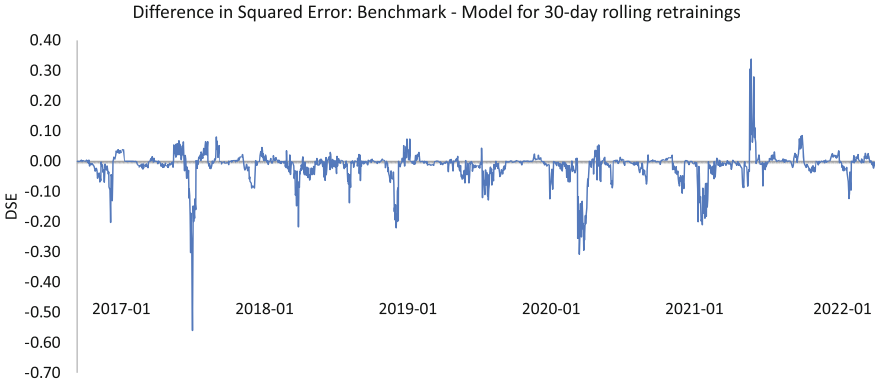


Fig. 8. Difference in squared errors of model compared to benchmark for 30-day rolling retraining.

Figure 9 shows the RMSE evaluated from each different starting point on the x-axis to the end of the dataset (May 2022). As we move to later starting points

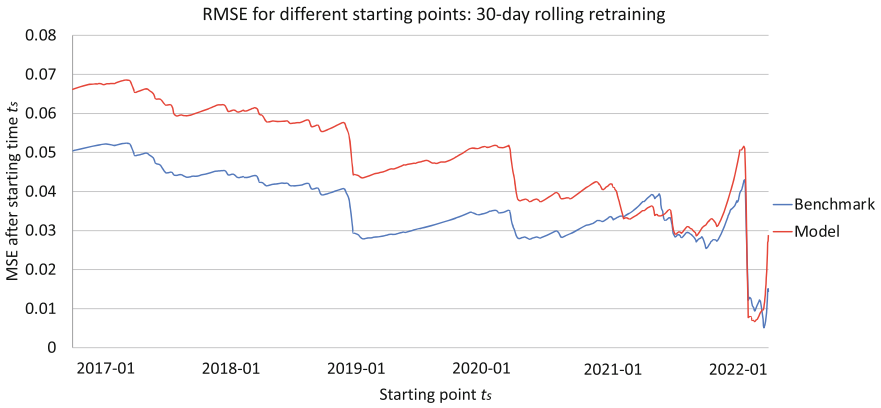


Fig. 9. Root mean squared errors of model compared to benchmark for different analysis starting points.

on the x-axis, it is worth noting that there is more training data incorporated into the model before the test set for RMSE. Toward the end of the dataset, the model becomes more competitive with the benchmark, surpassing it when measured over the final year of data.

Figure 10 shows a similar plot of RMSE evaluated from different starting points on the x-axis, but with the calculation restricted to the top 10% times of volatility. Here volatility is calculated as 24 h rolling volatility of hourly returns. The model is overall more competitive with the benchmark for top 10% volatility times compared to all times, and surpassing it by a sizable amount measured over the final year of data. Note that as suggested in Fig. 8, the outperformance of the model in the final year largely rests on a short period of high outperformance.

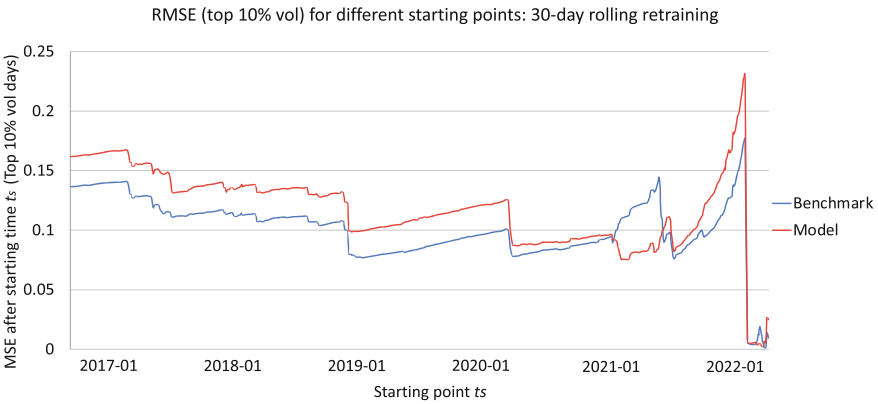


Fig. 10. Root mean squared errors over top 10% volatility times for different analysis starting points. Model compared to benchmark.

C Analysis of Celo PoS Data

In addition to Ethereum data, we also analyse data on the Celo PoS network (Figs. 11, 12, 13 and 14). This analysis involves some further features involving PoS systems as well as Celo's dual token model. This additionally serves as a first look at the analysis of a PoS system with historical data spanning longer than a year. In comparison, a similar analysis of Ethereum's new PoS system does not yet have enough history at the current time to perform a good analysis.

The price recovery is generally poorer than for the ETH/USD price explored earlier. This is likely explained by the higher volatility of Celo compared to Ethereum as well as the smaller size of historical data available.

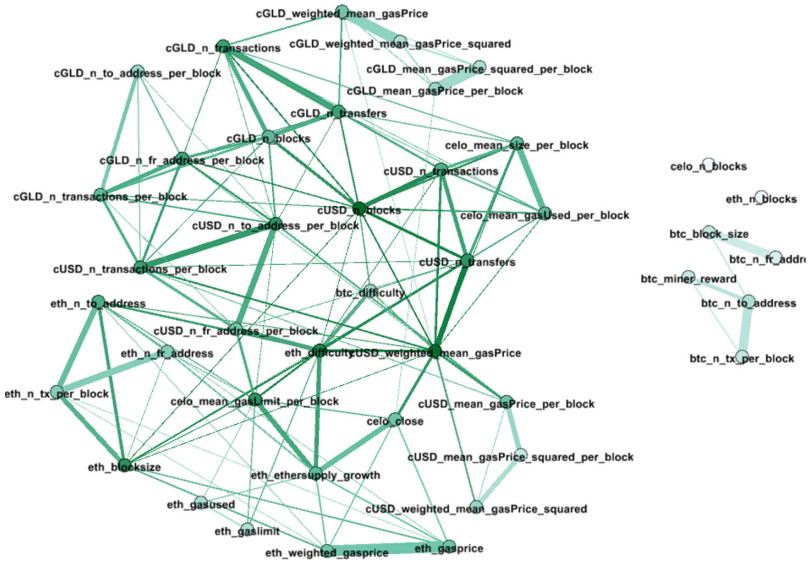


Fig. 11. Graphical network visualization from sparse inverse covariance estimation.

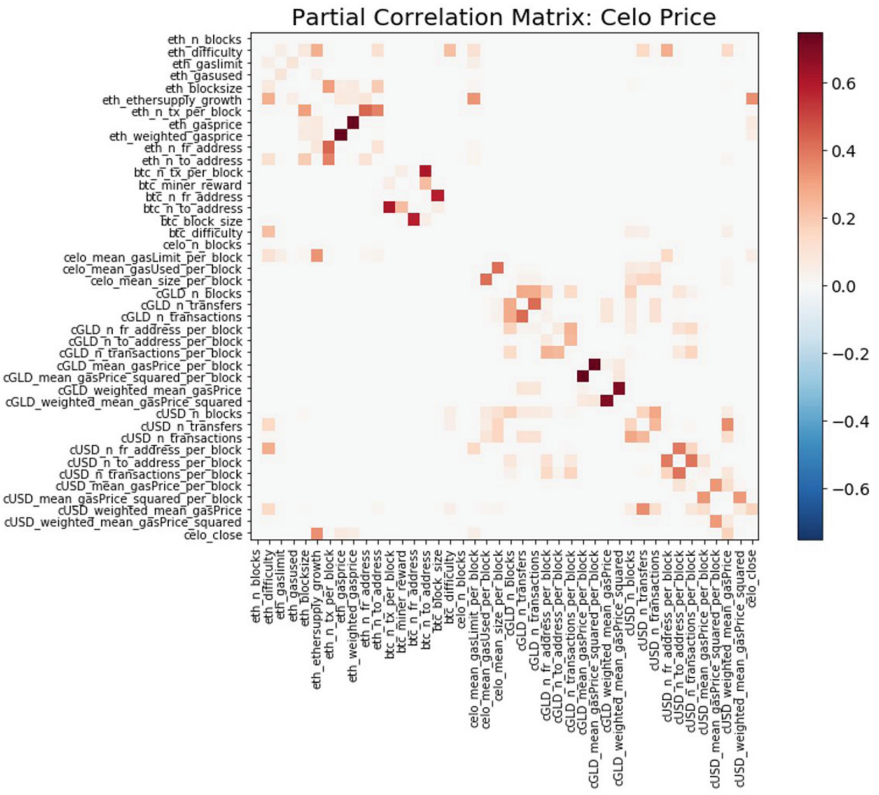


Fig. 12. Partial correlation matrix from sparse inverse covariance estimation.

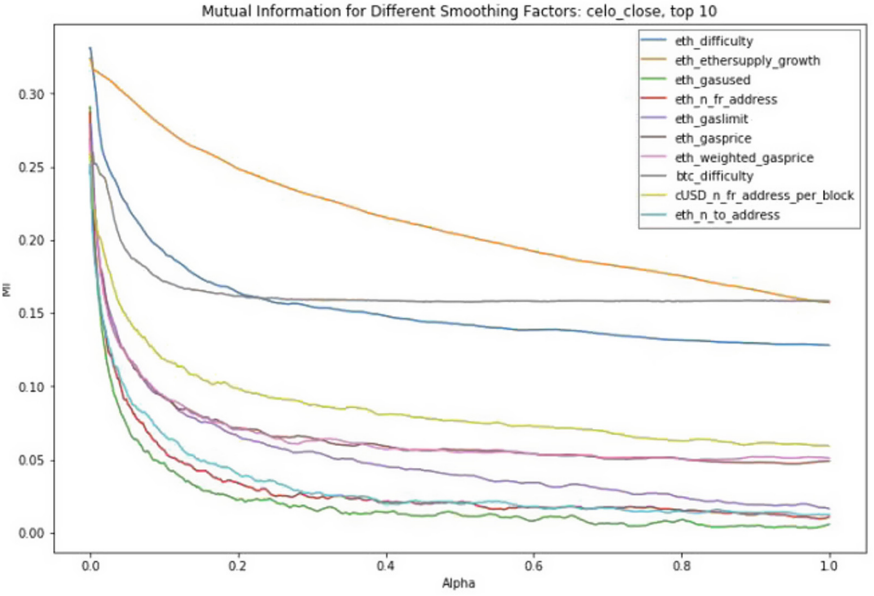


Fig. 13. Mutual information of price data and features, with smoothing α.

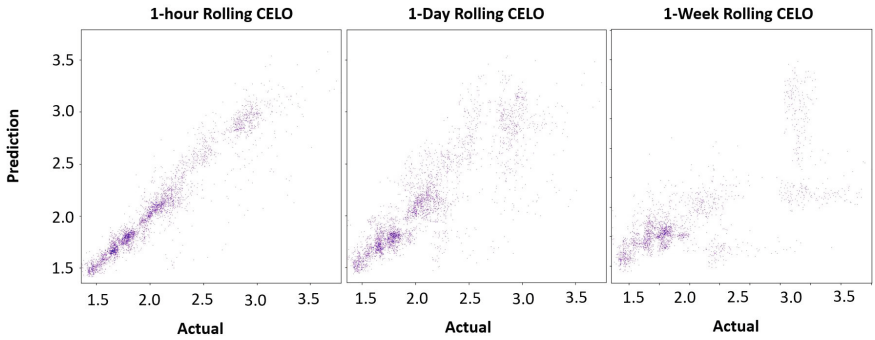


Fig. 14. Recovered price vs actual for random forest with given retraining periods.

References

1. Angeris, G., Chitra, T.: Improved price oracles: Constant function market makers. In: Proceedings of ACM Advances in Financial Technologies, pp. 80–91 (2020)
2. Athey, S., Parashkevov, I., Sarukkai, V., Xia, J.: Bitcoin pricing, adoption, and usage. Working Paper No. 3469 (17) (2016)
3. Buterin, V.: Blockchain Resource Pricing, pp. 1–32 (2018). https://ethresear.ch/uploads/default/original/2X/1/197884012ada193318b67c4b777441e4a1830f49.pdf
4. Easley, D., O'Hara, M., Basu, S.: From mining to markets: the evolution of bitcoin transaction fees (2018). https://doi.org/10.1007/s10551-015-2769-z

5. Easley, D., López de Prado, M., O'Hara, M., Zhang, Z.: Microstructure in the machine age. Rev. Financ. Stud. **34**(7), 3316–3363 (2021)
6. Fanti, G., Kogan, L.: Economics of proof-of-stake payment systems (2019)
7. Friedman, J., Hastie, T., Tibshirani, R.: Sparse inverse covariance estimation with the graphical lasso. Biostatistics **9**(3), 432–441 (2008)
8. Huberman, G., Leshno, J.D., Moallemi, C.: An Economic analysis of the bitcoin payment system*. SSRN Electron. J. 1–60 (2019)
9. Huo, L., Klages-Mundt, A., Minca, A., Münter, F.C., Wind, M.R.: Decentralized governance of stablecoins with closed form valuation. In: MARBLE (2022)
10. Intrinsic Technologies: Axiom: The ZK coprocessor for ethereum (2023). https://www.axiom.xyz/
11. Karantias, K., Kiayias, A., Zindros, D.: Smart contract derivatives. In: Mathematical Research for Blockchain Economy: 2nd International Conference MARBLE 2020, Vilamoura, Portugal, pp. 1–8. Springer (2020)
12. Klages-Mundt, A., Harz, D., Gudgeon, L., Liu, J.Y., Minca, A.: Stablecoins 2.0: economic foundations and risk-based models. In: ACM AFT, pp. 59–79 (2020)
13. Klages-Mundt, A., Schuldenzucker, S.: Design of the gyroscope consolidated price feed and circuit breaker system (2022)
14. Kroll, J.a., Davey, I.C., Felten, E.W.: The economics of bitcoin mining, or bitcoin in the presence of adversaries. WEIS 2013 (Weis), 1–21 (2013). Accessed 11–12 June 2013
15. Nicolas, H.: The economics of bitcoin transaction fees. SSRN Electron. J. (2014). https://doi.org/10.2139/ssrn.2400519
16. Prat, J., Benjamin, W.: An equilibrium model of the market for bitcoin mining. CesIfo Working Papers (January), p. 26 (2017)
17. Werner, S.M., Perez, D., Gudgeon, L., Klages-Mundt, A., Harz, D., Knottenbelt, W.J.: SOK: Decentralized finance (defi). In: ACM AFT (2022)



Exploring Decentralized Governance: A Framework Applied to Compound Finance

Stamatis Papangelou(✉), Klitos Christodoulou, and George Michoulis

Department of Digital Transformation, Business School, University of Nicosia, 46 Makedonitissas Avenue, CY-2417, P.O. Box 24005, 1700 Nicosia, Cyprus
papangelou.m@unic.ac.cy

Abstract. This research proposes a methodology which can be used for measuring governance decentralization in a Decentralized Autonomous Organization (DAO). DAOs, commonly, have the ambition to become more decentralized as time progresses. Such ambitions led to the creation of decentralized governance models that use governance tokens to represent voting power. Relevant research suggests that the distribution of the governance tokens follows centralized accumulations in a few wallets. By studying the accumulations of voting power from a DeFi protocol, this research presents a framework for identifying and measuring decentralization via analyzing all the various governance sub-systems instead of focusing on one or a small group. Governance within a DAO is a multi-layered process. By examining the decentralization of each layer or subsystem within the overarching governance structure, we can compose a comprehensive understanding of the entire protocol. To demonstrate this method, this paper uses the Compound Finance protocol as a case study. The first sub-system that this research discusses is the delegated and self-delegated wallets which are the only entities that can participate in the voting process in the Compound platform. The second sub-system is the actual proposals and votes that have taken place in the protocol's governance. Data is derived directly from the protocol's web data and for two time periods.

Keywords: DAOs · DeFi · DAO governance · Decentralization · Compound finance

1 Introduction

Decentralized finance (DeFi) is now an established term in the crypto ecosystem. Financial services that can be offered to both businesses and individuals in a decentralized manner are at the core of the DeFi ecosystem. Some of the provided services may include: exchanges, lending and borrowing, taxes, credit, insurance, etc. [3,23]. Decentralized Autonomous Organizations (DAOs) are at the core of most of the DeFi protocols [23]. As the name suggests, a DAO shall

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
P. Pardalos et al. (Eds.): MARBLE 2023, 2024.
https://doi.org/10.1007/978-3-031-48731-6_9

use decentralized ways to be operated and governed. Although the services and governance operations of the DeFi protocols rely purely on the decentralized ledger [7,22], the decentralization of the governance and decision-making process in the DeFi DAOs has to be questioned [1,14,20].

Decentralization is an aspect long discussed in the blockchain ecosystem. In Bitcoin and proof-of-work systems in general, decentralization of the nodes and hashing power is necessary to build trust and transparency in the ecosystem [24]. With the introduction of proof-of-stake and governance tokens in DAOs decentralization moved from "computer decentralization" to "economic decentralization". As it will be discussed in Sect. 3.1 most of the current research focuses on studying only one decentralization sub-system. Blockchain applications are formed from a "stack" of different sub-systems [21] forming a decentralized service to the end user. This "stack" starts from underlined ledger to the user interface and each sub-system connects with the next one unidirectional. If one sub-system is compromised then all the dependent ones above can be also compromised. Therefore, monitoring decentralization has to be done in the various sub-systems in order to provide answers to the whole platform.

1.1 Motivation

Decentralized autonomous organizations have adopted smart-contract-based governance structures. Instead of operating on a traditional founder, manager, and employee hierarchy base, in the majority of the operational DAOs, decisions on management and development of the organization are made by governance token holders and their delegates. As will be discussed in Sect. 3, current studies use decentralization monitoring methodologies similar to the ones used on mining nodes. The operations and consensus of DeFi protocols and in general DAOs, form more complete multidimensional decision-making systems in comparison with blockchain consensus protocols. This system requires a better understanding of the decentralization structure and the methodologies on which decentralization can be measured.

1.2 Contribution Summary

Research questions:

1. How can we analyze more decentralization dimensions in a DAO?
2. How do we monitor decentralization of a DAO via studying the different sub-systems?
3. How can we introduce the time dimension in the decentralization analysis?

This research will use one of the first governance token-based organizations named Compound as a case study on the proposed framework. Since all the well-established DAOs use such governance models to operate [17], the analysis of this organization will be used as a case study to formulate answers to the research questions. Compound was chosen as it provides accessible available data in their

font-end without the need to look at the on-chain data. Compound is a marketplace that allows individuals to borrow and lend digital assets. This platform is a decentralized protocol that is built on top of the Ethereum blockchain. All users can interact with the protocol's governance by voting and proposing via the COMP token delegations [4]. Decentralization of governance data, including voter delegations, proposal votes, and vote totals, will be analyzed quantitatively in this article.

Specifically, this paper will include in Sect. 2 the description of the Compound protocol and its governance methods. Section 3 shall contain the relevant work on the subject and the discussion of the given literature. It is the Methodology and Data overview that is being discussed in Sect. 4. And lastly, Sect. 4.2 may contain a description of the findings and Sect. 6 that will consist an applied use of the proposed framework will consist of a general discussion and future work.

2 Compound

The Compound protocol allows its users to stake (the process of "locking") up their crypto assets in order to earn interest or rewards and lend or borrow some of the supported assets such as: Ethereum, DAI, SAI, USD Coin, Tether, Augur, etc. [2]. This decentralized application allows anyone that owns such crypto assets to engage in the lending and borrowing process without the need to involve traditional financial services [8].

The main vehicle on which this protocol operates is the cTokens which are digital assets that represent the staked amount in the platform [8]. cTokens are "Ethereum Request for Comment"-20 (ERC-20) [5] Ethereum tokens that take the form of every underlined asset, such as cETH, cDAI, cREP, etc. These tokens are transferable and tradable due to their on-chain nature, therefore, other decentralized Applications (dApps) can support them. Interest is calculated and offered through the Compound protocol based on the available liquidity for each of the offered crypto assets on the platform. The rates are connected to the supply and demand of the market and they are constantly adjusted. For every 15 s each user's cTokens are increased by 1/2102400 which is equal to the fraction of 15 s blocks per year [2,4,8].

2.1 Governance

The Compound protocol can only be configured or upgraded by the COMP governance token holders or their delegations. Changes that go through the voting mechanism include collateral variables, interest rates, new markets, system characteristics, and so on. The COMP governance token has one-to-one voting power. COMP owners can delegate votes to themselves or any other Ethereum wallet [4]. Holders of COMP are unable to vote or create a proposal unless they have delegated their tokens to themselves or to another address [8]. The proposals are executable code that can modify the protocol and/or its operations. In order for an individual to create a proposal they need to have at least 60,000

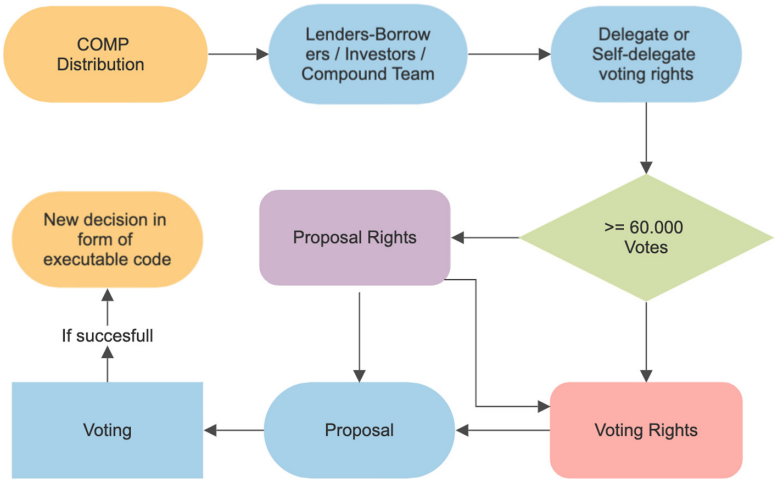


Fig. 1. Compound Governance Process

COMP delegated to their address [8]. If the total majority of votes surpass the minimum of 4% of total delegated COMP (i.e. 400,000 COMP) then the proposal is quoted in the Timelock smart contract that will stay at a minimum of 48 h before being implemented into the protocol [4,8,9]. This process is generally described in Fig. 1.

3 Relevant Work

The relevant literature in the area of Defi governance decentralization mainly focuses on the governance token distribution from major DeFi protocols. The [20] focus on the distribution of governance token from IDEX, MakerDAO, Compound, Curve, and Uniswap. The approach of the authors focuses on how the governance token distribution of all the holders is compared to the top 20 holders. The results suggest high levels of centralization of voting power across the protocols but the authors do not provide mathematical metrics to back their results. Reference [14] aims to also study the level of decentralization between DeFi protocols by using decentralization metrics. The study explains the governance models of Balancer, Compound, Uniswap, and Yearn Finance and derives data from Defipulse. The authors observe high amounts of centralization across the applications with the Gini Coefficient [6] taking values from 0.82 to 0.98 and Nakamoto Coefficients [15] from 82 to 9; with Compound being the most centralized platform. Their research provides good foundations on how to apply decentralization metrics in DeFi and what types of wallets sampling methods may exclude, but the limitation of the study is the absence of the time dimension and the delegation of votes. Moreover, [1] extends previous research by introducing new decentralization metrics that can be applied in DAOs and measures the governance decentralization of Uniswap, Maker, ShushiSwap, UMA,

and Yearn Finance. This research also introduces the time dimension by sampling their data with 6-month intervals and again the indexes indicate the high centralization of governance in the DeFi ecosystem. The most relevant study that follows similar approach to this paper is from [12]. The authors of the paper conducted an empirical study on the state of three prominent Decentralized Autonomous Organizations (DAO) governance systems on the Ethereum blockchain: Compound, Uniswap, and ENS. They used a comprehensive dataset of all governance token holders, delegates, proposals, and votes to analyze the distribution and use of voting power in these systems. The authors also evaluated the level of decentralization in these systems and studied the voting behavior of different types of delegates.

3.1 Literature Discussion

The majority of the previous studies have introduced methods for calculating decentralization on the governance token distributions from mainly Ethereum on-chain data on wallets that hold governance tokens. This method is excluding delegations and only covers the centralization dangers of the token holder subsystem. In the case of Compound, COMP token holders are not a solid part of governance; instead, their delegates are. Therefore, this is the main subsystem on which decentralization analysis can be conducted, alongside other important sub-systems below the delegates which are the proposals and actual votes that are submitted. Additionally, the importance of including the fundamental governance aspects of each individual DAO before conducting a decentralization analysis was not discussed in the given literature and can be a great subject of future research.

4 Methodology and Data

The methodology consists of data gathering from the official Compound Governance Website and then the presentation of existing decentralization metrics in order to find estimations on how decentralized is the governance of this Defi protocol. All the used code and data sets are uploaded and available on a GitHub[1] repository.

4.1 Data

This research relies on information manually scraped from the official Compound website [9]. The samples consist of two main sets, the first set is the top 100 leader-board based on their voting power and the individual proposals that were issued into their voting system. Due to the difficult sampling process and the distinct characteristic of this research is the leader-board sampling from two time periods (August 2021 and January 2022) and the decentralization monitoring of

[1] https://github.com/manospgl/COMPDecentralization.

the actual governance process on the proposals. Yet an other reason for the two sampling periods falls on the nature of blockchain itself. Because the token and delegate ownership exist on-chain then a clear time-window cannot exist, we can only extract a "snapshot" of the ledger as we do now. In order to get an extensive data sampling for such protocols a sampling framework must be used in order to get automated snapshots of the ledger. The data were manually extracted from the HTML website code found in Compound Governance page. The ambition for the DeFi protocols aim to become more decentralized over time, thus the data sample is derived from two time periods. Additionally, the proposals by nature are a recurrent process so they include the time dimension. The Compound leader-board consists of the address IDs, address names, votes (which translate to delegations owned), vote weight, and proposals voted on. Additionally, due to the already identified names of many of the addresses, extensive research on the internet on the identity of every address has been conducted and it's included on the leader-board data sets. The proposals include a total of 82 proposals from the initial proposal on May 2020 to January 2022. The samples contain the "for" and "against" address names together with their votes, the total "for" and "against" addresses that were involved in the proposal, the total "for" and "against" votes, the address that posted the proposal and the poll result as seen in Fig. 2.

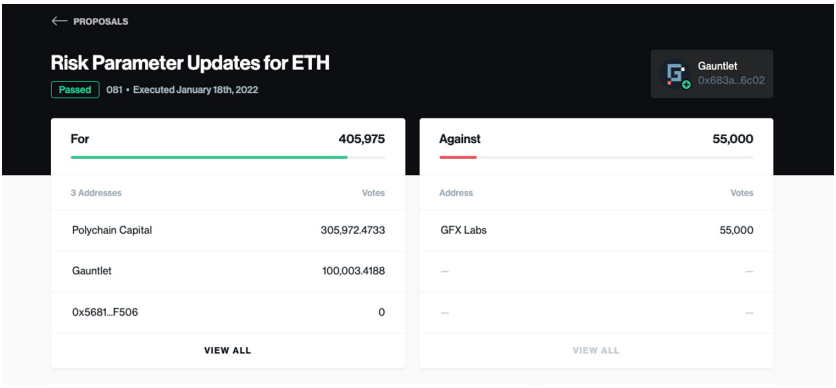


Fig. 2. Compound Proposal 081 screenshot

4.2 Methodology

The initial part of the proposed framework consist of the fundamental analysis of the various governance layers for given protocol, as demonstrated in Sect. 2 After identifying the distinct sub-systems, the goal is to measure the level of decentralization in all the systems via the use of existing decentralization metric coefficients, the Gini and Nakamoto. Gini Coefficient is used in economics as a measurement of statistical dispersion in order to represent the income or wealth inequality in a society [16]. The main purpose of this index is the measurement

of inequality among frequency distributions as delegations of governance tokens. This coefficient takes values from 0 to 1, with 0 representing the perfect equality and 1 the maximum inequality of the values [10]. Mathematically the Gini coefficient is based on the Lorenz curve which, in the case of token distributions, plots the proportion of the total tokens on the number of the population. This index is also mathematically defined by the half of the relative mean absolute difference which is equivalent to the Lorenz Curve definition [13]. In more detail, the Gini Coefficient is given by Eq. 1 [10].

$$Gini(u) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|u_i - u_j|}{2\sum_{i=1}^{n}\sum_{j=1}^{n}u_j} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|u_i - u_j|}{2n\sum_{j=1}^{n}u_j} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|u_i - u_j|}{2n^2\bar{u}} \quad (1)$$

where u_i is the total voting power (Governance token voting delegations) for each wallet i with a total of n wallets and $\bar{u}$ is the average in normalized scale [10].

On the other hand, Nakamoto Coefficient is defined as the minimum number of entities in a system or sub-system that obtain equal or more than 50% of the total capacity, this is calculated by aggregating the minimum of the subsystems [19]. Mathematically, the Nakamoto coefficient for a subsystem u with N entities with $j_1 > ... > j_N$ the proportions that each of the participants N controls in the subsystem such as $n : \sum_{i=1}^{n} j_i > \frac{1}{2}\sum_{i=1}^{N} j_i$, can be defined as seen in Eq. 2 [19].

$$N = \min\left(n : \sum_{i=1}^{n} j_i > \frac{1}{2}\sum_{i=1}^{N} j_i\right) \quad (2)$$

In the case of this study, the Nakamoto coefficient is the minimum number of entities (wallets) that can control more than 50% of the voting power.

The proposed decentralization analysis framework for Compound finance governance model consists of the following parts:

1. Distribution of usable governance power (e.g. delegated tokens), static and overtime.
   - Majority power distribution over the addresses that hold the majority of votes
   - Distribution tendencies over a time period
2. Qualitative characteristics of the dominant addresses
   - Identification of the dominant addresses
   - Identifying possible incentives according to the nature of the organisation or the individual behind the address
3. Proposal dominance distribution
   - Dominance on wallets that create proposals

   - Dominance on voting results
4. Used voting power analysis
   - Probabilities of desirable result from the power dominant addresses
   - Decentralization of the used power over past votes.

For analyzing the proposal dominance distribution and used voting power analysis the paper uses the voting data from the proposals. The proposing and result dominance is observed by the wallets that submitted the proposal and each voting result. In order to get the probabilities of desirable result from the dominant addresses this research used each of the dominant addresses voting history. The probability of a desirable result ($P_{desired}$) from a specific address i is determined by dividing the number of desirable results (votes for a proposal that succeeded or votes against a proposal that failed) by the total number of times that address i participated in voting, as seen in Eq. 3.

$$P^{i}_{desired} = \frac{\text{Number of desired results from address } i}{\text{Total vote participation of address } \ i} \tag{3}$$

Moreover, the last part of this analysis contains the study of the decentralization levels of the voters. The sampling of the proposal follows an assenting chronological order. In order go plot the possible decentralization trend, the research used the Gini and Nagamoto coefficients in the total votes that were submitted in each individual proposal.

Compound decentralization analysis using the proposed framework Considering that the fundamental sub-system structure of Compound has been discussed in Sect. 2, In this section, shall take the aforementioned data and do our own statistical research to get a better sense of how decentralized the Compound protocol's governance really is.

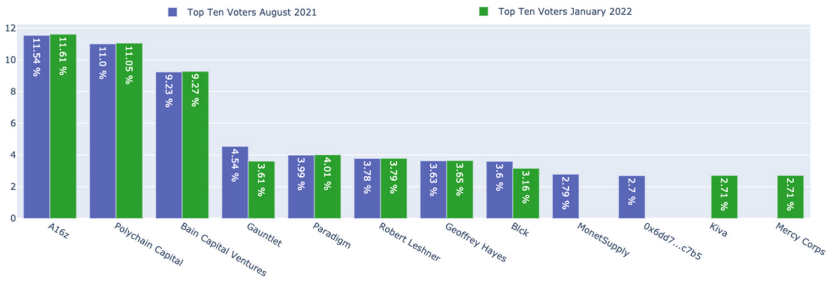


Fig. 3. Top ten voting dominance

4.3 Top 100 Leaderboard

The first step of the analysis for the voting leader-board data is to observe the power of the top vote delegated COMP holders. In detail, the voting power dominance of the top 10 wallets from the two time periods shall be presented.

Figure 3 shows the dominance of the top 10 addresses compared to the sum of the voting power of the entire leaderboard. What the graph shows, is the undeniable dominance of the top 10 voters on the token distribution, with 56,8% dominance on August 2021 and 55,5% on January 2022. It is obvious that in both cases the Nakamoto Coefficient is 8. The distribution of power suggests that if the top 8 wallets will like to form a consortium they can potentially influence any poll to their liking. What the data from the proposals should provide is how this voting power is actually been used.

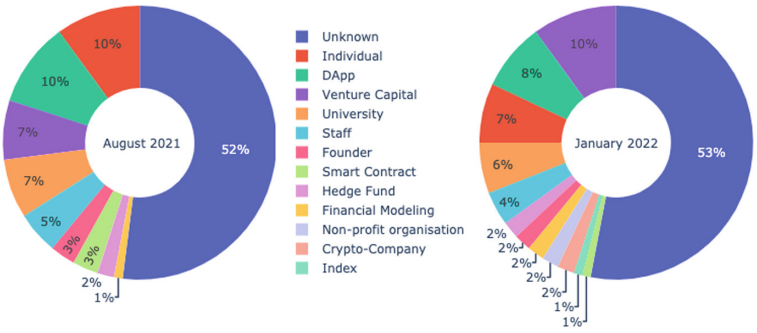


Fig. 4. Delegated Voters Category Distribution

Table 1. Distinct Address categories and their count of occurrence

Category	Count (August 2021)	Count (January 2022)
Venture capital	7	10
Hedge fund	2	2
Founder	4	2
Financial modeling	1	2
Unknown	52	53
Non-profit organisation	–	2
Individual	10	7
Crypto-company	–	2
DApp	10	8
University	7	6
Index	–	1
Smart contract	3	1
Staff	5	4

The next step of this research is to observe the demographics of the address owners. Applications built on the blockchain, including the blockchain technol-

ogy itself, advocate for privacy and independence [18]. One could argue that in a decentralized distribution system, only unknown persons will operate and influence a protocol, in keeping with the principle of anonymity and individuality. However, it is inefficient to begin building a platform and its laws solely via the efforts of individuals, especially in the absence of a centralized developing team to make the early judgments and funding. Therefore, most of the DeFi platforms start with centralized development and funding with the ambition to become decentralized overtime. While centralized entities like Venture Capitalists (VCs), founders, etc. may be entitled to a disproportionate share of the tokens at the outset, it is only fair to spread the initial governance token allocations to the early public users, the early investors, and the founding team.

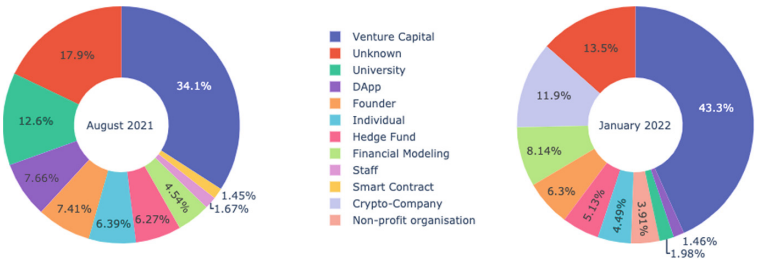


Fig. 5. Weighted Delegated Voters Category Distribution

Hence, this study is aiming at observing this transition. As was mentioned in Sect. 4.1 Compound provides some of the known names from the leaderboard. From individual research of every address, the data suggest the following distinct categories of the addressees. Table 1 shows all the individual categories for the addresses on the top 100 leader-board, their number of occurrences, and Fig. 4 presents visually their distribution. As mentioned before, it is important, in terms of decentralization, who owns the majority of the voting power. As Fig. 4 and Table 1 suggest that the majority of the voters, in terms of wallet population, are held by unknown individuals.

Also, the Compound governance model does not advocate for a scheme in which one wallet equals one vote in elections; rather, it proposes a mechanism in which one token equals one vote. Therefore, it is more valuable to study the distribution of wallet types weighed on their voting power. Although previously it is observed that the majority of the wallets have unknown origins if each category is weighed to their voting dominance then the distribution is significantly affected. Figure 5 shows that the majority of the voting power is held by Venture Capitals, and their total power has grown between August 2021 and February 2022 with the power of unknown wallets also shrinking in that period.

Lastly, the decentralization metrics for the top 100 voter leaderboard shall be analyzed. The numbers suggest that the Gini Coefficient for August 2021 is 0.764 and for January 2022 is 0.753. Additionally, Figs. 6 and 7 visualize the

Lorenz curve and outline the Nakamoto coefficient in both time periods is equal to 8. The Lorenz curve (solid line) represents the cumulative distribution of voting power among delegates, sorted in ascending order. The x-axis represents the cumulative proportion of delegates, while the y-axis represents the cumulative proportion of voting power. The line of equality (dashed line) represents a perfectly equal distribution of voting power. The deviation of the Lorenz curve from this line illustrates the degree of inequality in the distribution of voting power. The Nakamoto coefficient is represented by the vertical dashed line, indicating the minimum proportion of delegates that together hold more than 50% of the total voting power.

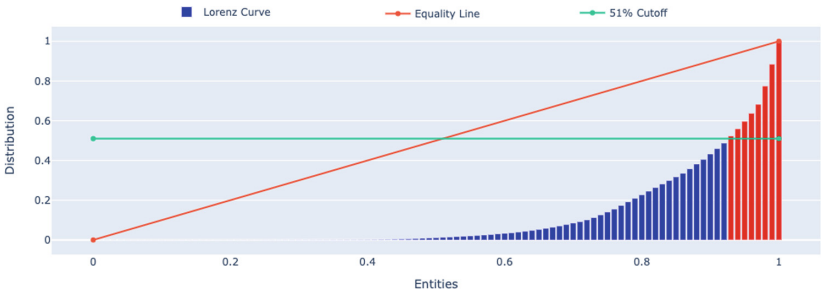


Fig. 6. August 2021 leaderboard

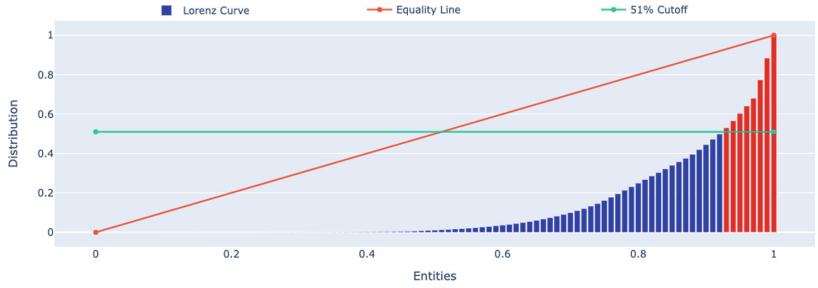


Fig. 7. January 2022 leaderboard

4.4 Proposals

At this stage, all the decentralization analysis is done in order to formulate what a selected few can potentially do due to their concentrated power. A similar analysis of the proposals shall provide how decentralized the power that has actually been used in the governance of this protocol is.

Figure 8 visualizes the distribution of the proposal results. The individual possible proposal results are passed, failed, and canceled. Figure 8 presents that the majority of the results are passed compared to the other two possible outcomes. Additionally, Fig. 8 shows the distribution of the wallets that have posted a proposal. It is observed that the majority of the proposals are posted by only three wallets. The lack of diversity of wallets posting proposals was expected due to the fundamental protocol requirement that a wallet must have more than 65.000 votes to post a proposal.

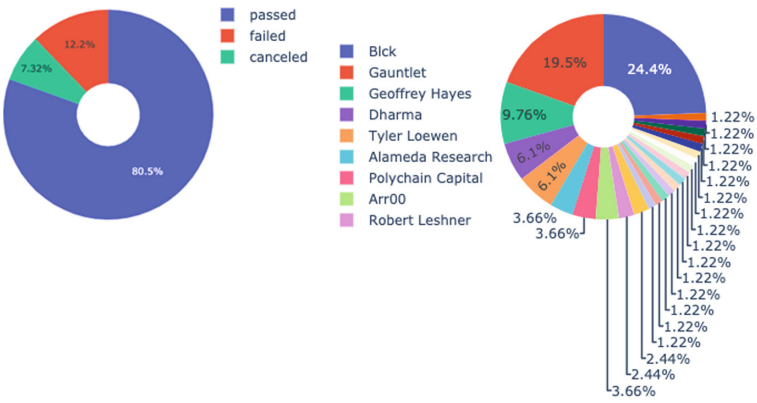


Fig. 8. Proposal Results and Domination of Wallets that Proposed

It is of high interest to observe the influence the majority holders obtain. In order to accomplish that, the study has taken the average probability of success (success is stated when the vote is equal to the poll result) on the two different vote options (for and against) and on their average. Figure 9 shows the probability of success for the top 10 wallets (From August 2021). This graph shows the major influence that the top 10 wallets have on the result of the proposal. For the positive votes, the top ten wallets have an average success probability of over 90%. It is observed that only seven out of ten wallets have voted negatively and their probability of success is much lower than the positive ones. Despite that, all the top 10 (with the exception of "Ox6dd7...c7b5" which haven't voted in any election) have probabilities of success over 80%.

This study will conclude by examining what the decentralization metrics suggest for the ideas. Because the proposals are a recurrent process it is suggested that all the individual proposal metrics shall be studied. For the Gini coefficient, the average is 0.768 with a Standard Deviation of 0.128. The Nakamoto coefficient average is 2.9 with a Standard Deviation of 1.031. It can be observed that the average Gini coefficient is similar levels as the top wallet leader-board, but a significant deviation on the Nakamoto coefficient at 2.9 compared to 8. The decline of the Nakamoto coefficient suggests that on average 2.9 wallets hold the voting majority. But as it has been discussed previously it is important to study whether the level of decentralization is either rising or declining. Figure 8

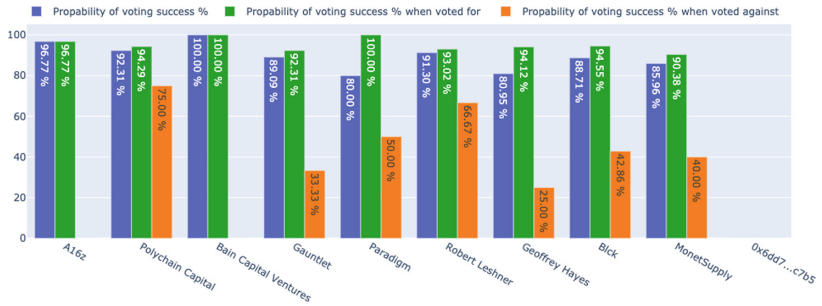


Fig. 9. Probabilities of Proposal Success form top 10 wallets

plots all the decentralization metrics from all the proposals. The proposals are sampled with assenting chronological order, thus providing a linear time dimension on the graph. From Fig. 10 it can be observed that indeed there is a small negative trend in the Gini coefficient, due to major vote holders involved over the last proposals, resulting in a similar observation with the leader-board with both sets having decentralization growth.

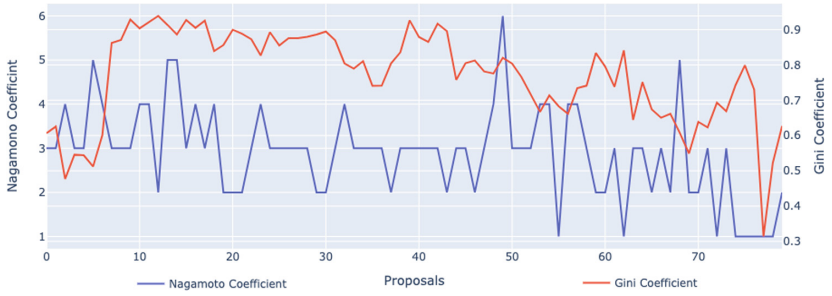


Fig. 10. Nagamoto and Gini coefficients timeline

We also observe some outliers in the proposals in both the coefficients. This is caused due to the nature of Gini and Nakamoto coefficient. Firty, both coefficient do not perform great as a decentralization metric when the sample population is small. This can be described with a short applied example: if in a single proposal they were only two address with exactly equal balance of votes this, then the Gini takes value of 1, stating full decentralization, and Nakamoto takes the value of 2 suggesting high levels of centralization (proof in Appendix A). Similar case to this example is observed on proposal 080 were only two wallets voted with 305969,8246 and 70014,3833 votes each, hence outliers values are observed in Fig. 2. Secondly, Nagmoto is by nature volatile because it only takes integer values compered to the Gini.

5 Discussion

The web data of delegates was more important than on-chain governance token allocations, as in prior publications. Decentralizing COMP holders may provide various consequences. The top delegated voters possess little or no COMP tokens, which is why the outcome would have been different. At the time of writing this article, Etherscan[2] and Compound's[3] website [9,11], state that the top delegated voter A16z's COMP token balance is zero. A16z has the most voting power, yet they have zero COMP tokens, and the other top 8 delegated voters (who have more than 50±% of the voting power) own only 50.9 of the 1,5 Million distributed COMP tokens. Thus, this DeFi protocol's decentralization study based on governance token distributions would be incorrect compared to the delegated wallets.

Furthermore, following the limitations of this work, the following questions are still open and can be included in future research. How each individual protocol can defer on the governance decentralization layering? What are the driven forces behind voting and voter interactions? What are the limits of the decentralization indexes?

Additionally, future research should also focus on delegation mapping and processes. A rigorous study of all governance-related protocol transactions will be used to map delegates, proposals, and votes. Future research may apply similar methods to a bigger sample of DeFi ecosystem DAOs and beyond.

6 Conclusions

This study proposes a new framework for evaluating decentralization levels in a Decentralized Autonomous Organizations. The Compound DeFi lending/borrowing protocol was used as a case-study in order for the authors to apply their methodology. Web-scraping in the governance web interface of the Compound platform over two time periods provided the data. Venture Capital businesses and other private organizations have a significant voting power concentration. In both time periods, the delegated voter leaderboard was analyzed using Gini and Nakamoto decentralization coefficients. The above criteria were applied to all votes submitted in proposals through January 2022. Results indicate significant voting success in the top ten delegated voters with Gini indexes of 0.764 and 0.753 for August 2021 and February 2022, respectively, and Nakamoto coefficients of 8. Finally, the decentralization levels of votes filed into proposals with an average Gini coefficient of 0.768 and average Nakamoto coefficient of 2.9, both indexes decreasing. The aforementioned statistics indicate that the delegated votes and proposal subsystems have low decentralization but are trending toward higher decentralization.

[2] https://etherscan.io/address/0x9aa835bc7b8ce13b9b0c9764a52fbf71ac62ccf1.
[3] https://compound.finance/governance/address/0x9aa835bc7b8ce13b9b0c9764a52fbf71ac62ccf1.

Appendix A Proof for Gini and Nakamoto Coefficients When $n = 2$ and $u_1 = u_2$

Gini coefficient using Eq. 1:

$$Gini = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} |u_i - u_j|}{2n^2 \bar{u}} \tag{4}$$

With $n = 2$ (the number of addresses) and $\bar{u} = v$ (the mean of votes for each address), we get:

$$Gini = \frac{|v - v| + |v - v|}{2 \times 2^2 v} \tag{5}$$

The term in the numerator becomes 0 (because $|v - v| = 0$), hence:

$$Gini = 0 \tag{6}$$

This confirms our previous intuition that the Gini coefficient would be 0 in this case, indicating perfect equality.

Nakamoto Coefficient using Eq. 2:

$$N = \min \left(n : \sum_{i=1}^{n} j_i > \frac{1}{2} \sum_{i=1}^{N} j_i \right) \tag{7}$$

Here, j_i represents the sorted list of addresses by votes in decreasing order. But since we only have two addresses and each has an equal amount of votes, the list is either $[v, v]$ or $[v, v]$ depending on how you sort it.

If we take $n = 1$, then $\sum_{i=1}^{n} j_i = v$, and $\frac{1}{2} \sum_{i=1}^{N} j_i = v$. Since the two quantities are equal, $n = 1$ does not satisfy the condition of n being the minimum such that $\sum_{i=1}^{n} j_i > \frac{1}{2} \sum_{i=1}^{N} j_i$. Thus, we have to take $n = 2$, which satisfies the condition, because $\sum_{i=1}^{2} j_i = 2v$, and $\frac{1}{2} \sum_{i=1}^{N} j_i = v$.

So we get:

$$N = 2 \tag{8}$$

This indicates that it would take both addresses to reach a majority of the voting power. This is consistent with our previous explanation that the Nakamoto coefficient would be 2 in this case.

References

1. DeFi, Not So Decentralized: The Measured Distribution of Voting Rights, HICSS, vol. 55. ScholarSpace is the institutional repository for the University of Hawai at Mānoa and is maintained by Hamilton Library (2022)
2. Academy, B.: What is compound finance in defi? (9 2020), https://academy.binance.com/en/articles/what-is-compound-finance-in-defi

3. Andrei-Dragos, P.: Decentralized finance (defi) -the lego of finance. Soc. Sci. Edu. Res. Rev. **349**, 314–341 (2020). https://www.researchgate.net/publication/343054092_DECENTRALIZED_FINANCE_DEFI_-THE_LEGO_OF_FINANCE
4. Bavosa, A.: Building a governance interface (4 2020), https://medium.com/compound-finance/building-a-governance-interface-474fc271588c
5. BitcoinWiki: ERC20 Token Standard - Ethereum Smart Contracts - BitcoinWiki – en.bitcoinwiki.org. https://en.bitcoinwiki.org/wiki/ERC20 (2021), [Accessed 29-Jan-2023]
6. Ceriani, L., Verme, P.: The origins of the gini index: extracts from variabilita e mutabilita (1912) by corrado gini. J. Econ. Inequality **10**, 421–443 (2012)
7. Christodoulou, P., Christodoulou, K.: A decentralized voting mechanism: engaging erc-20 token holders in decision-making. In: 2020 Seventh International Conference on Software Defined Systems (SDS), pp. 160–164. IEEE (2020)
8. Compound: Compound docs (2022). https://compound.finance/docs
9. Compound: Compound governace (2022). https://compound.finance/governance
10. Dorfman, R.: A formula for the gini coefficient. The Review of Economics and Statistics, pp. 146–149 (1979)
11. Foundation, E.: Etherscan (2021). https://etherscan.io/
12. Fritsch, R., Müller, M., Wattenhofer, R.: Analyzing voting power in decentralized governance: Who controls daos? (2022). arXiv:2204.01176
13. Gastwirth, J.L.: A general definition of the lorenz curve. Econometrica: J. Econ. Soc. 1037–1039 (1971)
14. Jensen, J.R., von Wachter, V., Ross, O.: How decentralized is the governance of blockchain-based finance: Empirical evidence from four governance token distributions (2021). arXiv:2102.10096
15. Kusmierz, B., Overko, R.: How centralized is decentralized? comparison of wealth distribution in coins and tokens. In: 2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS), pp. 1–6. IEEE (2022)
16. Milanovic, B.: A simple way to calculate the gini coefficient, and some implications. Econ. Lett. **56**(1), 45–49 (1997)
17. Morrison, R., Mazey, N.C., Wingreen, S.C.: The dao controversy: the case for a new species of corporate governance? Front. Blockchain **3**, 25 (2020)
18. Sims, A.: Blockchain and decentralised autonomous organisations (daos): the evolution of companies? alexandra sims* (the final version of this article is forthcoming in the new zealand universities law review). 28 New Zealand Universities Law Review 96, 1–29 (2019). https://doi.org/10.2139/ssrn.3524674
19. Srinivasan, B.S., Lee, L.: Quantifying decentralization (7 2017). https://news.earn.com/quantifying-decentralization-e39db233c28e
20. Stroponiati, K., Abugov, I., Varelas, Y., Stroponiatis, K., Jurgelevicience, M., Savanth, Y.: Decentralized Governance in DeFi: Examples and Pitfalls (2020). https://static1.squarespace.com/static/5966eb2ff7e0ab3d29b6b55d/t/5f989987fc086a1d8482ae70/1603837124500/defi_governance_paper.pdf
21. Touloupou, M., Christodoulou, K., Inglezakis, A., Iosif, E., Themistocleous, M.: Benchmarking blockchains: The case of xrp ledger and beyond. In: Proceedings of the Fifty-third Annual Hawaii International Conference on System Sciences, (HICSS 55). HICSS, vol. 55, p. 8. ScholarSpace is the institutional repository for the University of Hawai at Mānoa and is maintained by Hamilton Library (2022)
22. Vlachos, A., Christodoulou, K., Iosif, E.: An algorithmic blockchain readiness index. In: Multidisciplinary Digital Publishing Institute Proceedings, vol. 28, p. 4 (2019)

23. Werner, S.M., Perez, D., Gudgeon, L., Klages-Mundt, A., Harz, D., Knottenbelt, W.J., Calcaterra, C., Kaal, W.A.: Decentralized finance (defi). SSRN Electron. J. 1–19 (2021). https://doi.org/10.2139/ssrn.3782216
24. Zhang, R., Xue, R., Liu, L.: Security and privacy on blockchain. ACM Comput. Surv. (CSUR) **52**(3), 1–34 (2019)

A Mathematical Approach on the Use of Integer Partitions for Smurfing in Cryptocurrencies

Bernhard Garn(✉), Klaus Kieseberg, Ceren Çulha, Marlene Koelbing, and Dimitris E. Simos

MATRIS Research Group, SBA Research, Vienna 1040, Austria
{bgarn,kkieseberg,mculha,mkoelbing,dsimos}@sba-research.org

Abstract. In this paper, we propose the modelling of patterns of financial transactions - with a focus on the domain of cryptocurrencies - as splittings and present a method for generating such splittings utilizing integer partitions. We further exemplify that, by having these partitions fulfill certain criteria derived from financial policies, the splittings generated from them can be used for modelling illicit transactional behavior as seen in smurfing.

Keywords: Money laundering · Smurfing · Integer partition

1 Introduction

Money laundering (ML) is a global societal threat that affects all countries to varying degrees with criminal organizations and terrorist groups moving funds equivalent to billions of US dollars each year through the international financial system. The term ML describes the process by which the proceeds of crime and the true ownership of those proceeds are concealed or made opaque so that they appear to come from a legitimate origin [8]. Criminals do this by disguising the *sources*, changing the *form* or moving the funds to a *place* where they are less likely to attract attention. The emergence of digital currencies has given rise to *crypto laundering*, which is the use of cryptocurrencies and their specific properties for ML [12]. To combat ML in fiat currencies, the EU has designed directive 2015/849,[1] which includes legislation defining fixed thresholds above which transactions are to be regulated; with similar regulations being planned on an EU level for crypto-currencies as of this writing.

In finance, *smurfing* is a term describing a common technique used by money launderers to conceal illicit activities by breaking up noticeable transactions into smaller inconspicuous ones in order to avoid detection. A simplified example visualizing smurfing is shown in Fig. 1: black proceeds are separated into transactions from wallet A to four different recipients B, C, D and E and finally collected in a

[1] https://eur-lex.europa.eu/eli/dir/2015/849/oj.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
P. Pardalos et al. (Eds.): MARBLE 2023, 2024.
https://doi.org/10.1007/978-3-031-48731-6_10

single wallet F again. Smurfing is considered to be a major threat to the integrity of financial systems as it makes the detection of illicit financial transactions a significant challenge for authorities. The increase in these concerns has also led to an increase in the importance of anti-money laundering (AML) transactions monitoring systems, whose primary purpose is to identify potential suspicious behavior within legitimate transactions.

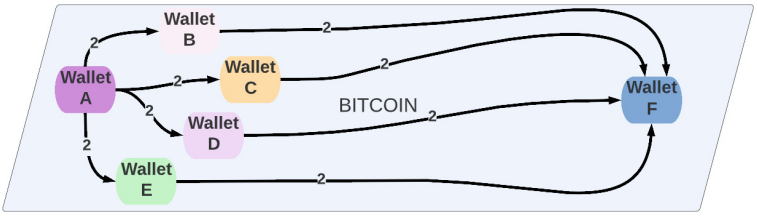


Fig. 1. Smurfing.

In this paper, we present a modelling approach for smurfing in the case of cryptocurrencies using combinatorics, more specifically, integer partitions (IPs). Derived from this model, we provide a constructive approach for splitting up large and conspicuous financial transactions, with the goal that the resulting sets of smaller transactions offer enhanced capabilities for avoiding detection in a real world financial setting. Our approach considers as input a specific amount in some currency and outputs one or all possible splittings of this amount while taking into account a set of criteria derived from regulations, such as thresholds or properties indicating suspiciousness according to monitoring practices. In other words, our aim is to construct a set of splittings respecting certain criteria with the intention to give rise to transactions able to bypass regulations.

This paper is structured as follows. In Sect. 2, we discuss related work. We propose our mathematical approach on the use of IPs for smurfing in Sect. 3. We conclude and indicate an outlook on future work in Sect. 4.

2 Related Work

Several studies have explored transaction monitoring for AML efforts, including the work of [13], in which the authors developed a novel approach for detecting and disrupting ML activities using a combination of network-based clustering and classification techniques. Other works [2–4,7] considered monitoring systems to identify potential suspicious behaviors embedded in legitimate transactions. A framework for detecting ML networks is presented in [9]. In [10], the authors present an effective method for detecting suspicious smurf-like subgraphs in ML activities. A visualization system for analyzing Bitcoin wallets is described in [11], aiding investigators in tracking the flow of Bitcoin and identifying potential criminal activities. Blockchain analysis platforms have been proposed in [5,6],

which offer support for the investigation of monetary flows as well as the identification of transactions having specific properties.

3 Using Integer Partitions to Create Patterns for Smurfing

In this section, we present the details of our proposed combinatorial modelling approach for certain transactional behaviors with potential subsequent use in ML. More precisely, we consider how to determine constraints and then subsequently construct a complete split of a given amount of money (possibly from illicit sources) into multiple, smaller transactions which are designed to avoid detection.

Mathematical notation for IPs, sequences and sets. Firstly, as we will extensively use IPs for modelling certain transaction behaviors with potential links to ML, we consider the following mathematical definition of an IP:

Definition 1. [cf. [1]] A *partition* of a positive integer n is a finite non-increasing sequence of positive integers $(\lambda_1, \lambda_2, \ldots, \lambda_r)$ such that $\sum_{i=1}^{r} \lambda_i = n$. The λ_i are called the *parts* of the partition and r is called the *length* of the partition.

To illustrate the concept of IPs and the corresponding notation, we give a simple example:

Example 1. The integer 5 has the following 7 partitions: $(1,1,1,1,1)$, $(2,1,1,1)$, $(2,2,1)$, $(3,1,1)$, $(3,2)$, $(4,1)$ and (5).

Secondly, we use the symbol $\mathbb{N}$ to denote the positive integers and the mathematical notation $\mathbb{N}^{<\mathbb{N}}$ to denote the *set of finite sequences of natural numbers*, i.e.,

$$\mathbb{N}^{<\mathbb{N}} = \{p \mid p \text{ is a finite sequence of natural numbers}\}.$$

Further, we write $[\mathbb{N}^{<\mathbb{N}}]^{<\mathbb{N}}$ to denote the *set of finite sets of finite sequences of natural numbers*:

$$[\mathbb{N}^{<\mathbb{N}}]^{<\mathbb{N}} = \{A \mid A \text{ is finite and } \forall p \in A\, p \text{ is a finite sequence of natural numbers}\}.$$

We illustrate the above defined notions with the following

Example 2. As examples of elements of $\mathbb{N}^{<\mathbb{N}}$ we consider $(3,5,85)$, $(9,2,11,1,1)$ and $(5,5,5,5,5,5,5)$. If we now take the set consisting of these three finite sequences, we get $\{(3,5,85),(9,2,11,1,1),(5,5,5,5,5,5,5)\} \in [\mathbb{N}^{<\mathbb{N}}]^{<\mathbb{N}}$.

3.1 Problem Formulation

The transaction patterns caused by smurfing are the consequence of splitting up an initial (large) amount of money via multiple transactions. Therefore, IPs as basis for a mathematical model of smurfing are a natural choice.

Problem: Given an amount of money in some currency and some regulations, find all IPs of the corresponding integer with properties such that the resulting transactions are not affected by applicable regulations.

The input to this problem contains, for the scope of this work, the initial total amount of money expressed in the smallest unit of the used currency,[2] applicable monitoring regulations on transactions in the corresponding currency as well as additional properties of the desired splittings like the number of (or bounds on) the total number of resulting transactions or the sizes of the occurring individual parts. We consider as solution, for the scope of this work, a splitting of the given amount fulfilling the listed requirements given in the problem input.

To tackle this problem, we need to find a function which takes as input an amount of money (in an arbitrary currency) and gives as output possibilities to transfer the money split up into one or more transactions.

$$\text{amount} \mapsto \text{ possible splitting(s) for smurfing of the amount}$$

Each amount (expressed in the indivisible unit of the currency) gives a natural number and each possible splitting can be expressed as a finite sequence of natural numbers summing up to the original amount. This means we are looking for a function S such that for each $n \in \mathbb{N}$ and each $p \in S(n)$ we have[3] $\sum_{i=1}^{\text{length}(p)} p(i) = n$. In other words, we are looking for a function S such that for each $n \in \mathbb{N}$, each $p \in S(n)$ is an IP of n:

$$\begin{aligned} S\colon\ & \mathbb{N} \to [\mathbb{N}^{<\mathbb{N}}]^{<\mathbb{N}} \\ & n \mapsto \{p \mid p \text{ is an IP of } n\}. \end{aligned}$$

To obtain functions which offer more modelling capabilities with potential advantages for smurfing, additional requirements on $S(n)$ can be added. The impact of a specific given IP on the corresponding smurfing is that its encoded *splitting* determines the size and number of transactions involved. Here, several practical aspects have to be taken into account: For instance, if the size of the individual parts of the IP is too small, it might result into too many transactions, making it difficult to perform them. On the other hand, if the parts are too large, they may attract unwanted attention and increase the risk of detection. Hence,

[2] We assume the existence of a smallest currency unit; in the real world, this assumption holds—in particular—for the US$, for the Euro and for Bitcoin.

[3] Each element p of $S(n)$ is a finite sequence, we write $p = (p(1), p(2), \ldots, p(\text{length}(p)))$.

considering various properties of IPs leads to precise modelling capabilities for the resulting transactions. In Table 1, we provide a mapping of these properties to the application domain of smurfing, and give a more detailed explanation below:

Table 1. Translation table for smurfing and discrete mathematical model via IPs.

Smurfing	Integer partition
Total money value	Integer
Number of transactions	Length of the IP
Transaction amount	Part of the IP
Minimal/maximal value of transaction amounts	Minimal/maximal size of parts of IP
Difference between transaction amounts	Difference between parts of IP

Total money value: The total money value is expressed in terms of the underlying indivisible unit of the currency.

Number of transactions: The given amount of money is split into smaller parts with each part corresponding to one transaction.

Transaction amounts: The individual parts of the IP determine the amount of the respective transactions.

Minimal/maximal value of transaction amounts: Constraints on admissible IPs give lower and upper bounds on transaction amounts.

Difference between transaction amounts: Variance in transaction amounts can be enforced by specifying a range for the differences between neighbouring parts in an IP.

3.2 Possible Implications of the Conceptualization of Smurfing via Integer Partitions

There are restrictions on the usable IPs for smurfing to ensure that the resulting transactions encoded in the IP are *likely to go undetected* by any *controlling entities*. In particular, the number of parts, the respective size of the parts, and the similarity between the parts are parameters to consider when trying to avoid detection, but all of these additional requirements on the admissible IPs reduce the number of possible splittings for the smurfing process of a given amount. We remark that if transactions above some value have to be reported to some *monitoring entity*, then this can be avoided by using IPs with smaller parts than this threshold; however, this could result in a larger number of transactions, which could also be conspicuous. Stating these conditions in form of the function we are looking for, gives the requirements to restrict the length of the partitions to at most ℓ, for some fixed $\ell \in \mathbb{N}$ and to restrict the size of the parts of the

partitions to at most k, for some fixed $k \in \mathbb{N}$. In a more formal notation, we have:

$$S_k^\ell\colon \quad \mathbb{N} \to [\mathbb{N}^{<\mathbb{N}}]^{<\mathbb{N}}$$
$$n \mapsto \{p \mid p \text{ is an IP of } n,\ \text{length}(p) \leq \ell \text{ and for each } i \leq \text{length}(p)\ p(i) \leq k\}$$

To decide the minimal possible amount, potential transaction fees could be taken into account: if the amount is too small, then large parts of the amount might be lost due to the transaction fees. In general, to avoid suspicion by monitoring, the transaction amounts should not be classified as outliers. Further, to obscure the connection between the transactions, it could be useful to choose amounts which are not too similar to each other.

Using our proposed mathematical formulation, our models are *constructive*, in the sense that mathematical software can be used to actually construct IPs respecting desired properties, when given corresponding parameter values, and hence compute the selected version of the function S. In our proposed methodology for modelling smurfing with IPs, the steps of the corresponding procedure to use the advantages of IPs for potential smurfing are as follows:

1. Select the amount in some currency that should be split in the smurfing and express it in the indivisible unit of the used currency, yielding an integer n.
2. Determine the requirements for the smurfing to circumvent controls to avoid detection based on applicable regulations.
3. Translate identified requirements from the previous step into properties of IPs according to Table 1.
4. Use a suitable computer algebra system to compute the IPs of n satisfying the required properties.
5. If there are no IPs of n with the needed properties, weaken or change some of the requirements, and repeat the process from step 3 onwards.
6. If an IP of n fulfilling the requirements has been computed, select the recipients of the individual parts and then arrange the transactions.

The last step of this procedure has to be considered in detail, since the choice of recipients, transaction times, and the routing of the transactions can significantly impact the overall conspicuousness.

3.3 Examples for Integer Partitions and Smurfing

In this section, we give two examples illustrating how mathematical software can be used to compute IPs satisfying specific restrictions motivated by requirements for smurfing patterns. All presented computations were carried out in SageMath[4] (version 9.3) on a laptop (Intel i5-10210U, 1.60 GHz base with 2.11 GHz boost clock and 16GB of RAM). In SageMath, IPs can be computed by the function

[4] https://github.com/sagemath/sage-windows/releases/tag/0.6.3-9.3.

`Partitions()`, which has various parameters on which we rely extensively in our examples.

In the examples that we give in this section, we work with Bitcoin (BTC) as currency. For BTC, the underlying indivisible unit is called Satoshi, where an amount of 1 BTC is equal to $100,000,000$ Satoshis. Hence, each smurfing process of n BTC corresponds to an IP of the natural number $n \cdot 100,000,000$.

Basic example for IPs and smurfing Here, we consider an amount of 0.00001BTC for splitting via IPs for subsequent potential use in smurfing. Taking into account that the smallest unit of BTC is *one Satoshi* and their ratio, we obtain that $0.00001\text{BTC} = 1,000$ Satoshis. Hence, we are interested in IPs of $1,000 \in \mathbb{N}$. To this end, applying the function `Partitions(1000).cardinality()` in SageMath, yields that in total there are around $24 \cdot 10^{30}$ IPs of $1,000$. As mentioned before, not all IPs are equally well suited. Assume that in order to bypass certain regulations and to fulfill other requirements, it was determined that the total number of transactions should be between 5 and 20, and that each transaction amount should be between 10 and 200 Satoshis. To obtain only partitions of $1,000$ respecting these two constraints, more parameters of the function `Partitions` in SageMath can be used:

```
x=iter(Partitions(1000, min_part=10, max_part=200, min_length=5,
                                                    max_length=20))
```

Following up with repeated invocation of the function `next(x)` gives IPs respecting the parameters. The first three given IPs are the following[5]:

$$[200, 200, 200, 200, 200], [200, 200, 200, 200, 190, 10], [200, 200, 200, 200, 189, 11].$$

Example using constrained IPs and smurfing We now consider additional requirements on the transactions, yielding further restrictions on the admissible IPs. We assume that the analysis of regulations and other considerations led to the requirements that all individual transaction amounts in the generated splitting have to have roughly the same size, but still all need to be different from each other. SageMath provides the parameters `max_slope` and `min_slope` for the function `Partitions`, which determine the maximal and minimal difference between two consecutive parts of an IP. More precisely, for an IP $[\lambda_1, \ldots, \lambda_r]$ the maximum and the minimum of the values $\lambda_{i+1} - \lambda_i$, for $1 \leq i < r$, are restricted[6]. Specifically, we now also require that the absolute value of the difference between any two neighbouring transaction amounts is at least 10 and at most 50 Satoshi. We can obtain the resulting admissible transaction amounts by using the following command of SageMath:

```
x=iter(Partitions(1000, min_part=10, max_part=200, min_length=5,
                    max_length=20, max_slope=-10, min_slope=-50))
```

[5] Here, we adopt the notation from SageMath and denote sequences as $[\lambda_1, \ldots, \lambda_r]$.
[6] Note that it follows from Definition 1 that we have $\lambda_{i+1} - \lambda_i \leq 0$, for all $1 \leq i < r$.

Invoking the function `next(x)` twice, we obtain the following two IPs:

$$[200, 190, 180, 170, 155, 105], [200, 190, 180, 170, 154, 106].$$

Any one of the partitions obtained by this computation fulfills the stated requirements above and can be selected as a **specific** *money distribution template* for use within smurfing, where the resulting transactions would be similar to what is shown in Fig. 1. Using the last IP given above, a total amount of $1,000$ *Satoshi* is transferred from one wallet (corresponding to Wallet A in Fig. 1) via six transactions with amounts of 106, 154, 170, 180, 190 and 200 *Satoshi*, respectively, to receiving wallets (corresponding to Wallets B, C, D and E in Fig. 1).

4 Conclusion and Future Work

In this paper, we presented a mathematical modelling approach on how to split illicit financial transactions into smaller transactions to avoid detection in cryptocurrencies using IPs. In order to advance our initial mathematical approach further towards a constructive methodology, some questions in regards of the instantiation of parameters have to be studied; such as the implication of the order in which the transactions happen, how the respective recipients are chosen, or what is the optimal timing and routing for the transactions.

Acknowledgements. SBA Research (SBA-K1) is a COMET Center within the COMET - Competence Centers for Excellent Technologies Programme and funded by BMK, BMAW, and the federal state of Vienna. The COMET Programme is managed by FFG. Moreover, this work was performed partly under the following financial assistance award 70NANB21H124 from U.S. Department of Commerce, National Institute of Standards and Technology.

References

1. Andrews, G.E.: The Elementary Theory of Partitions. Encyclopedia of Mathematics and its Applications, pp. 1–15. Cambridge University Press (1984)
2. Demetis, D.S.: Fighting money laundering with technology: a case study of Bank X in the UK. Decis. Support Syst. **105**, 96–107 (2018)
3. Gao, S., Xu, D., Wang, H., Wang, Y.: Intelligent Anti-Money Laundering System. In: 2006 IEEE International Conference on Service Operations and Logistics, and Informatics, pp. 851–856 (2006)
4. Han, J., Huang, Y., Liu, S., Towey, K.: Artificial intelligence for anti-money laundering: a review and extension. Digit. Finance **2**, 211–239 (2020)
5. Haslhofer, B., Stüt z, R., Romiti, M., King, R.: GraphSense: A General-Purpose Cryptoasset Analytics Platform (2021)
6. Kalodner, H., Möser, M., Lee, K., Goldfeder, S., Plattner, M., Chator, A., Narayanan, A.: BlockSci: Design and applications of a blockchain analysis platform. In: 29th USENIX Security Symposium (USENIX Security 20). pp. 2721–2738. USENIX Association (2020). https://www.usenix.org/conference/usenixsecurity20/presentation/kalodner

7. Kingdon, J.: AI fights money laundering. IEEE Intell. Syst. **19**(3), 87–89 (2004)
8. Levi, M.: Money laundering and its regulation. Ann. Am. Acad. Pol. Soc. Sci. **582**(1), 181–194 (2002)
9. Mehmet, M., Wijesekera, D., Fuentes, M.: Money laundering detection framework to link the disparate and evolving schemes. J. Digit. Forensics, Sec. Law (JDFSL) 1558-7223 **8**, 41 (01 2013)
10. Starnini, M., et al.: Smurf-based anti-money laundering in time-evolving transaction networks. In: Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track, pp. 171–186. Springer International Publishing, Cham (2021)
11. Sun, Y., Xiong, H., Yiu, S.M., Lam, K.Y.: Bitanalysis: a visualization system for bitcoin wallet investigation. IEEE Trans. Big Data **9**(2), 621–636 (2023)
12. Wang, H.M., Hsieh, M.L.: Cryptocurrency is new vogue: a reflection on money laundering prevention. Sec. J. (01 2023)
13. Weber, M., et al.: Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019)



Bigger Than We Thought: The Upbit Hack Gang

Qishuang Fu[1], Dan Lin[2], and Jiajing Wu[2(✉)]

[1] School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China
fuqsh6@mail2.sysu.edu.cn

[2] School of Software Engineering, Sun Yat-sen University, Zhuhai 519082, China
lind8@mail2.sysu.edu.cn, wujiajing@mail.sysu.edu.cn

Abstract. The prosperous development of Ethereum has bred many illegal activities by malefactors, such as Ponzi schemes, theft of funds from exchanges, and attacks on service providers. Aiming to expedite the realization of their gains, criminals will launder illicit money, making it as difficult as possible for security companies or agencies to recover those illicit funds. In this paper, we focus on a typical security event on Upbit exchange and explore the scale of the gang behind the security event. Specifically, We construct a rough suspicious money laundering transaction network by crawling downstream transactions of 815 accounts marked as Upbit hacks. Then, in order to refine a more accurate gang of Upbit hacks, we design a suspiciousness indicator for money laundering and modify an existing general risk assessment framework based on propagation models to assess the money laundering risk of accounts. Based on the risks, we acquire an accurate gang for Upbit hacks. In the end, we find that the size of the Upbit hack gang is much bigger than we thought. We also present several interesting analyses of the Upbit hack gang.

Keywords: Ethereum · Upbit hack · Money laundering · Suspiciousness indicator

1 Introduction

Ethereum is the largest blockchain platform that supports smart contracts [1], and the popularity of Ethereum has sparked a significant increase in investment activity. Numerous investors register the account without cost on Ethereum to make transactions with Ether [2] (the native cryptocurrency on Ethereum) and tokens [3] (other cryptocurrencies on Ethereum). To date, the market capitalization of Ethereum has reached about 214 billion USD.[1]

However, due to the inadequate law enforcement on Ethereum, criminals are drawn to participating in illicit activities. Once the addresses of criminals

[1] https://coinmarketcap.com/currencies/Ethereum/.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
P. Pardalos et al. (Eds.): MARBLE 2023, 2024.
https://doi.org/10.1007/978-3-031-48731-6_11

acquire ill-gotten gains, the criminals will make every effort to launder money to cash out. The goal of money laundering (ML for short) in cryptocurrency is to move funds to addresses where the original criminal source can not be detected, and eventually to some service providers (such as exchanges, mixers, and loaning pools) that allow cryptocurrency to be exchanged for cash. More than 66.2 billion USD worth of cryptocurrency from 2017 to 2022 has been laundered by criminals [4].

ML is rampant, but the accounts used by illegal organizations for ML are generally concealed and unknown. For example, disreputable Upbit hacks stole 342,000 Ether from the Upbit exchange on November 27, 2019. There are only 815 addresses labeled as *Upbit hack* by a prominent blockchain explorer XBlock.[2] By examining the transaction records of those labeled accounts, it can be seen that those illegal accounts have transferred ill-gotten money to other accounts, finally laundering the illegal money to the service providers. It can be inferred that more than just those 815 labeled accounts are involved in the illegal organization's financial activities.

In this paper, we construct a rough suspicious ML transaction network by crawling downstream transactions of 815 accounts marked as Upbit hacks. Then, to refine a gang of Upbit hacks, we design an ML suspiciousness indicator for accounts and modify an existing general risk assessment framework based on propagation models to calculate the ML risks of accounts. Based on ML risks, we filter out a gang for Upbit hacks. In the end, we find that the size of the Upbit hack gang is much bigger than we thought. We also present several interesting findings of ML in the Upbit hack gang.

The multi-faceted contributions of the paper are summarized as follows:

- **A new ML suspiciousness indicator for accounts**: We have designed an indicator called *ML_Suspic* that measures the degree of suspiciousness of an account's involvement in ML by combining the characteristics of account ML on Ethereum.
- **A complete ML dataset of Upbit hacks**: We crawled and constructed a rough suspicious ML transaction network and refine the gang of Upbit hacks. Researchers can conduct more in-depth research based on this dataset. Our dataset is open-sourced at https://www.dropbox.com/scl/fo/8j1otjrnsa5b019pbp2xy/h?dl=0&rlkey=jivsf9ymaenzfj47to17fi5dd.
- **Insights on ML in Upbit hack gang**: We analyze the risk distribution of the Upbit hack gang and share some analyses of layering accounts, service providers, and a local ML process in the gang.

2 Rough ML Network Construction

To obtain the entire ML gang of the Upbit hacks, the first step is to construct a rough ML network. We crawl transaction records of the relevant accounts by a heuristic method. Then, we construct those accounts and transactions into a rough ML network.

[2] https://www.xblock.pro/cloud.

2.1 Crawling Tool and Event

The Upbit hack event that occurred on November 27, 2019, serves as a prominent example of ML on Ethereum, in which about 342,000 Ether was stolen. The perpetrators used a large number of accounts to transfer the funds to various service providers in layers, eventually being able to withdraw the funds. We have collected the dataset of Upbit hacks by utilizing a popular blockchain explorer XBlock and an open-source crawler toolkit called BlockchainSpider [5]. Within XBlock's Label Word Cloud, there are 815 accounts that have been labeled as related to the Upbit hacks incident.

2.2 Account and Transaction Data Crawling

To obtain the ML dataset, we utilized a heuristic method approach. ML generally consists of two types of transactions: layering transactions and integration transactions [6]. Layering transactions refer to the process of transferring dirty money to other accounts in order to obscure the funds' source through a layer-by-layer method. Integration transactions refer to the process of transferring illicit money to service providers, such as an exchange or a mixing service, intending to withdraw legal money.

To be specific, first, we utilized BlockchainSpider to crawl transaction records of 815 accounts that have been labeled as related to the Upbit hacks incident. We filtered out transactions that occurred before the incident and were traded with tokens. Then, we classified transaction records into layering transactions and integration transactions, based on whether the receiving accounts were categorized as service providers, such as wallets and loaning pools. On the one hand, integration transactions, which indicate successful ML, are identified when the receiving accounts are classified as service providers. On the other hand, transactions involving non-service providers are considered layering transactions and suggest that the ML process is still ongoing. We continued to collect transferring-out transactions from non-service providers until we arrived at integration transactions.

2.3 Network Construction

After obtaining accounts and transactions, we modeled them as a multi-directed weighted transaction network $G_{rough}= (N, E)$, where N and E represent the set of all accounts and transactions respectively. Each transaction is represented as a four-tuple (u, v, a, t), which means that at timestamp t, account u transfers Ether in the amount of a to account v, where $u, v \in N$. In the G_{rough}, there are more than two hundred thousand accounts and eight hundred thousand transactions, which is a huge gang. Among the transactions, there are five hundred thousand layering transactions and three hundred thousand integration transactions.

3 ML Network Refinement

In order to obtain a more accurate ML gang, we refine the rough network by quantifying the ML risk of accounts and removing accounts whose ML risk is below a threshold and related transactions from G_{rough}.

Quantifying the ML risks of accounts mainly involves two parts. The first part is to design an ML suspiciousness indicator for accounts based on ML characteristics [7,8]. The second part is to combine the suspiciousness indicator with a risk assessment framework [9] to calculate the ML risks of accounts in the rough ML network.

3.1 Design an ML Suspiciousness Indicator

ML accounts usually have two characteristics: (i) zero-out accounts, and (ii) fast-in and fast-out accounts:

Characteristic 1. ML accounts are zero-out middle accounts. Most received money from ML accounts will be transferred out, which leads to the balance of ML accounts being close to zero [7].
Characteristic 2. ML accounts are fast-in and fast-out accounts. Most of those accounts transfer money in and out within a short interval [8].

Based on those two characteristics, we designed an ML suspiciousness indicator for accounts to measure the degree of suspiciousness of the accounts' involvement in ML. We measure the ML suspiciousness of accounts by calculating the balance of accounts (which reflects whether accounts are zero-out middle accounts) and the average rate of in-out transaction pairs (which reflects whether accounts are fast-in and fast-out accounts).

- **Balance of an account**

The balance of an account n is defined as

$$B(n) = In(n) - Out(n), (n \in N),$$

where $In(n)$ represents the total amount of Ether received by n and $Out(n)$ is the total amount of Ether moved out from n.

- **Average rate of transaction pairs of an account**

The average rate of transaction pairs of an account n is defined as

$$A(n) = \frac{\sum T(e_{pair}(n))}{N(e_{pair}(n))}, (e_{pair} \in E),$$

where $e_{pair}(n)$ is an in-out transaction pair of n, $T(e_{pair}(n))$ is the interval between incoming and outgoing transactions of an in-out transaction pair of n, and $N(e_{pair}(n))$ is the number of transaction pairs of n. For instance, one Ether is transferred into n, and after one minute n transfers it to another account. Those two transactions are a $e_{pair}(n)$. One minute is a $T(e_{pair}(n))$.

• **ML suspiciousness of an account**

We have calculated $B(n)$ and $A(n)$ to measure whether the account has the characteristics 1. and 2. respectively, but the range of $B(n)$ and $A(n)$ are different. In order to standardize the range of those two indicators, We first make use of a logarithmic scale to transform the values of those two indicators, which helps to narrow the gap between values, and then apply min-max normalization to make those indicators in the range [0, 1].

The Normalized $B(n)$ is represented as

$$Nor_B(n) = \frac{\log(B(n)+1)}{\log(Max_B+1)}, (n \in N),$$

where Max_B is the maximum value of $B(n)$ in G_{rough}, the reason why we shift the indicator values to the right by one unit is to make $Nor_B(n)$ not smaller than zero.

Similar to $Nor_B(n)$, the Normalized $A(n)$ is represented as

$$Nor_A(n) = \frac{\log(A(n)+1)}{\log(Max_A+1)}, (n \in N).$$

Then, we use the arithmetic mean of $Nor_B(n)$ and $Nor_A(n)$ to calculate the ML suspiciousness. The ML suspiciousness of an account n is defined as

$$ML_Suspic(n) = \frac{1}{2}(Nor_B(n) + Nor_A(n)), (n \in N), \quad (1)$$

$ML_Suspic(n)$ ranges from 0 (low suspicious) to 1 (high suspicious).

3.2 Calculate ML Risks of Accounts

To calculate ML risks of accounts, we combined the ML suspiciousness with the Riskprop model [9]. By incorporating the ML suspiciousness into the Riskprop model and modifying the model, we acquire an ML risk for each account. The pseudo-code of ML risk calculation is described in Algorithm 1.

The ML risk ranges from 0 (very low risk) to 10 (very high risk). We have divided the risks into four levels based on values of risks, which are low risk ([0, 4]), moderate risk ([4, 6]), moderate-high risk ([6, 8]), and high risk ([8, 10]). We believe that accounts with a risk level below 4 and an average transaction amount of accounts less than 0.01 Ether are innocent accounts, and we remove those innocent accounts and corresponding transactions from G_{rough}.

4 Results and Analysis

After calculating ML risks of accounts in G_{rough}, we remove innocent accounts and corresponding transactions from G_{rough}. Finally, we acquire a $G_{refined}$ which is consist of 7,914 accounts and 109,572 transactions, which is a huge gang.

Algorithm 1 Accounts MonL Risk Calculation Algorithm

Input: Multi-directed Weighted Transaction Network $G_{rough} = (N, E)$
Output: MonL Risks of accounts
Initialize $trustiness^0 = 0.5$, $reliability^0 = 0.7, t = 0, \Delta = 1$
Calculate $MonL_suspiciousness$ of accounts using $MonL_suspic(n) = \frac{1}{2}(Nor_B(n) + Nor_A(n))$
Calculate $score$ of transactions using $score(u, v) = \frac{1}{2}(\frac{\log MN^{out} - 2\log|N^{out}(u)|}{\log MN^{out}} + \frac{\log MN^{in} - 2\log|N^{in}(v)|}{\log MN^{in}})$
while $\Delta \geq 0.01$ **do**
  $t = t + 1$
  Update $trustiness$ of payees using $T(v) = \frac{\sum_{u \in \mathrm{In}(v)} R(u) \cdot S(u,v) + MonL_suspic(v)}{|\mathrm{In}(v)|}$
  Update $reliability$ of payers using $R(u) = \frac{\sum_{v \in \mathrm{Out}(u)} T(u) \cdot S(u,v) + MonL_suspic(u)}{|\mathrm{Out}(u)|}$
  $\Delta_T = \sum_{v \in V} |T^t(v) - T^{t-1}(v)|$
  $\Delta_R = \sum_{u \in U} |R^t(u) - R^{t-1}(u)|$
  $\Delta = \max\{\Delta_T, \Delta_R\}$
end while
Calculate ML Risks of accounts using $Risk(u) = (1 - R(u)) \times 10$
return $Risk$

As shown in Fig. 1, we have analyzed an ML risk distribution of accounts in $G_{refined}$ and obtained the following observations and analyses: (1) In the entire refined ML network $G_{refined}$, there are 6,459 high-risk accounts far exceeding the 815 accounts labeled as Upbit hacks, which indicates that the $G_{refined}$ has added more suspicious accounts related to ML on the basis of the original 815 Upbit hacks. (2) There are a few moderate-risk accounts in $G_{refined}$, those moderate-risk accounts usually transferred money out only once, while moderate-high-risk and high-risk accounts usually transfer money out many times, since those moderate-risk accounts did not have a strong intention to launder the money actively, the ML risks are relatively low.

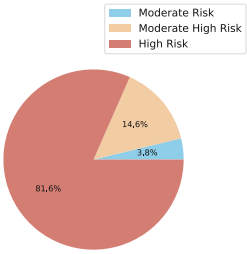


Fig. 1. Risk distribution of accounts in Upbit hack gang. The colors show the different levels of risk.

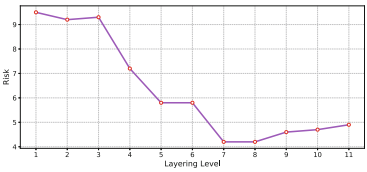


Fig. 2. The minimum risk of different layering level accounts.

As shown in Fig. 2, since we crawled transactions layer by layer, we categorized layering accounts into 11 layers and found that as the number of layers increased, the minimum risk of layering accounts decreased. That indicates that layering transfers of illicit money are indeed a method of risk diversification. It is important to prevent the layering transfer of accounts as early as possible.

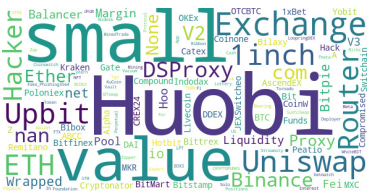


Fig. 3. Service providers of the Upbit hack gang.

The word cloud of service providers in the Upbit hack gang is shown in Fig. 3, we can tell that the stolen money from Upbit hacks flowed into centralized exchanges (such as Huobi and OKEx) and decentralized exchanges (such as Uniswap), and the stolen money also were swapped into other tokens (such as BTC and NFT).

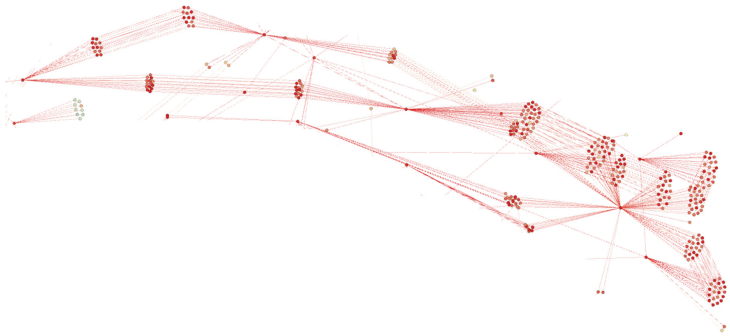


Fig. 4. The portion of ML process of Upbit hack gang. The darker the color of the node, the higher the risk.

As shown in Fig. 4, we present a local ML process of $G_{refined}$. The entire ML process is a multi-directional chain-like structure, while locally it takes on a diverging tree-like structure, which indicates that the members of the Upbit hack gang were very cunning. Not only did they layer-by-layer transfer the illicit money, but they also dispersed the black money to different accounts, greatly increasing the concealment of criminals.

5 Conclusion

In this paper, we explore the real scale of the Upbit hack gang. We construct a rough suspicious ML transaction network. Then we design an ML suspiciousness indicator for accounts and modify a general risk assessment framework to calculate ML risks of accounts. Based on ML risks, we acquire a more accurate gang of Upbit hacks. In the end, we find that the size of the Upbit hack gang is much bigger than we thought. We also present several interesting findings of ML in the Upbit hack gang.

Acknowledgements. The work described in this paper is supported by the National Natural Science Foundation of China (61973325).

References

1. Lin, D., Chen, J., Wu, J., Zheng, Z.: Evolution of ethereum transaction relationships: toward understanding global driving factors from microscopic patterns. IEEE Trans. Comput. Soc. Syst. **9**(2), 559–570 (2022)
2. Wu, J., Liu, J., Zhao, Y., Zheng, Z.: Analysis of cryptocurrency transactions from a network perspective: an overview. J. Netw. Comput. Appl. **190**, 103139 (2021)
3. Chen, W., Zhang, T., Chen, Z., Zheng, Z., Lu, Y.: Traveling the token world: a graph analysis of ethereum erc20 token ecosystem. In: Proceedings of The Web Conference, pp. 1411–1421 (2020)
4. Chainalysis Team: The chainalysis 2023 crypto crime report (2023). https://go.chainalysis.com/2023-crypto-crime-report.html
5. Wu, Z., Liu, J., Wu, J., Zheng, Z.: Transaction tracking on blockchain trading systems using personalized PageRank (2022). arXiv:2201.05757
6. Kolachala, K., Simsek, E., Ababneh, M., Vishwanathan, R.: Sok: money laundering in cryptocurrencies. In: Proceedings of the 16th International Conference on Availability, Reliability and Security, pp. 1–10 (2021)
7. Sun, X., Feng, W., Liu, S., Xie, Y., Bhatia, S., Hooi, B., Wang, W., Cheng, X.: Monlad: money laundering agents detection in transaction streams. In: Proceedings of the ACM International Conference on Web Search and Data Mining, pp. 976–986 (2022)
8. Sun, X., Zhang, J., Zhao, Q., Liu, S., Chen, J., Zhuang, R., Shen, H., Cheng, X.: Cubeflow: money laundering detection with coupled tensors. In: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 78–90 (2021)
9. Lin, D., Wu, J., Fu, Q., Zheng, Z., Chen, T.: Riskprop: account risk rating on Ethereum via de-anonymous score and network propagation (2023). arXiv:2301.00354

Staking Pools on Blockchains

Hans Gersbach[1], Akaki Mamageishvili[2], and Manvir Schneider[3]

[1] ETH Zurich and KOF Swiss Economic Institute, Zürich, Switzerland
[2] Offchain Labs, Zürich, USA
[3] Cardano Foundation, Zug, Switzerland

Abstract. On several proof-of-stake blockchains, agents engaged in validating transactions can open a pool to which others can delegate their stake in order to earn higher returns. We develop a model of staking pool formation in the presence of malicious agents and establish existence and uniqueness of equilibria. We then identify the potential and risks of staking pools. First, allowing for staking pools lowers blockchain security. Yet, honest stake holders obtain higher returns. Second, by choosing welfare optimal distribution rewards, staking pools prevent that malicious agents receive large rewards. Third, when pool owners can freely distribute the returns from validation to delegators, staking pools may disrupt blockchain operations, since malicious agents attract most delegators by offering generous returns.

Keywords. Delegation, Staking Pool, Blockchain, Governance.

© The Editor(s) (if applicable) and The Author(s), under exclusive license
to Springer Nature Switzerland AG 2023
P. Pardalos et al. (Eds.): MARBLE 2023, 2024.
https://doi.org/10.1007/978-3-031-48731-6