3rd International Conference on Blockchain Economics, Security and Protocols

Tokenomics 2021, November 18–19, 2021, New York University, USA (Virtual Conference)

Edited by

Vincent Gramoli
Hanna Halaburda
Rafael Pass

OASICS

Editors

Vincent Gramoli
University of Sydney, Australia
EPFL, Switzerland
vincent.gramoli@sydney.edu.au

Hanna Halaburda
NYU Stern School of Business, USA
hh66@stern.nyu.edu

Rafael Pass
Cornell University, Ithaca, USA
rafael@cs.cornell.edu

ACM Classification 2012
Theory of computation → Distributed algorithms; Security and privacy → Cryptography; Mathematics of computing → Mathematical analysis

ISBN 978-3-95977-220-4

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-220-4.

Publication date
March, 2022

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

Digital Object Identifier: 10.4230/OASIcs.Tokenomics.2021.0

ISSN 1868-8969

https://www.dagstuhl.de/oasics

Contents

Preface

The papers in this volume were presented at Tokenomics 2021, the 3rd International Conference on Blockchain Economics, Security and Protocols, hosted virtually at New York University from November 18th to 19th, 2021.

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

The Program Committee of Tokenomics 2021 was divided in two sub-committees. One sub-commitee of 26 international computer science experts co-chaired by Vincent Gramoli and Rafael Pass and another sub-committee of 36 international economics experts chaired by Hanna Halaburda. The Program Committee received 41 submissions, each submission was classified by their authors depending on its topic:

- 22 submissions were classified as mostly economics papers;
- 11 submissions were classified as overlapping both fields of computer science and economics;
- 8 submissions were classified as mostly computer science papers.

Each paper was assigned 3 reviewers among the program committee members based on the expertise and expressed interest of the members.

Out of all submissions, 5 papers were accepted to be presented and published in the proceedings of the conference and 13 papers were accepted for presentation only. Each submitted paper went through a reviewing process before being discussed online by the program committee until convergence towards a decision upon whether to include it in the program.

Lin William Cong (Cornell University, USA), Xi Li (Newcastle University, UK), Ke Tang (Tsinghua University, China) and Yang Yang (Tsinghua University, China) received the Best Paper Award of Tokenomics 2021 for their paper *Crypto Wash Trading*. The paper presents tests based on statistical and behavioral trading patterns to detect fake transactions on 29 cryptocurrency exchanges. These tests led to identify that an average of 70% of the trading volume on these exchange platforms is dedicated to wash trading, i.e., simultaneously selling and buying the same financial assets to create artificial activity in the marketplace. This work finds applications in forensics finance and suggests the potential of regulation in the area of cryptocurrency. The award was offered by the Blockchain & Platform Chair of the École Polytechnique.

Last but not least, the program also included three great keynote talks: Joe Halpern (Cornell University, USA) presented *Distributed Computing Meets Game Theory: Fault Tolerance and Implementation with Cheap Talk*, David Parkes (MIT, USA) presented *Mechanism Design for the Blockchain Transaction-Fee Market* and Catherine Tucker (Harvard University, USA) presented *Privacy and Blockchain.*

We are grateful to the sponsors of Tokenomics 2021: Blockchain & Platform Chair of the École Polytechnique and New York University, NY, USA.

Vincent Gramoli, Hanna Halaburda, and Rafael Pass

List of Authors

Lin William Cong (10)
Samuel Curtis Johnson Graduate School of Management, Cornell University SC Johnson College of Business, Ithaca, NY, USA

Daniel Engel (13)
Computer Science Department, Brown University, Providence, RI, USA

Ittay Eyal (3, 4)
Technion, Haifa, Israel; IC3

Matheus V. X. Ferreira (6)
Computer Science, Harvard University, Boston, MA, USA

Rowena Gan (9)
Information Technology and Operations Management Department, Cox School of Business, Southern Methodist University, Dallas, TX, USA

Joseph Y. Halpern (1)
Cornell University, Ithaca, NY, USA

Runchao Han (2)
Monash University, Melbourne, Australia; CSIRO-Data61, Melbourne, Australia

Maurice Herlihy (13)
Computer Science Department, Brown University, Providence, RI, USA

Charles M. Kahn (7)
University of Illinois at Urbana-Champaign, IL, USA

Roman Kozhan (11)
Warwick Business School, University of Warwick, Coventry, UK

Michael Junho Lee (8)
Federal Reserve Bank of New York, NY, USA

Xi Li (10)
Newcastle University Business School, UK

Zhichun Lu (2)
Cryptape, Hangzhou, China

Antoine Martin (8)
Federal Reserve Bank of New York, NY, USA

Daniel J. Moroz (6)
Computer Science, Harvard University, Boston, MA, USA

Takeshi Nakai (5)
The University of Electro-Communications, Tokyo, Japan

Serguei Netessine (9)
Operations, Information and Decisions Department, The Wharton School, University of Pennsylvania, Philadelphia, PA, USA

David C. Parkes (6)
Computer Science, Harvard University, Boston, MA, USA

Kazumasa Shinagawa (5)
Ibaraki University, Ibaraki, Japan; National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Alexander Spiegelman (3)
Novi Research, Herzliya, Israel

Mitchell Stern (6)
EECS, University of California at Berkeley, CA, USA

Ke Tang (10)
Tsinghua University Institute of Economics, Beijing, China

Robert M. Townsend (8)
Massachusetts Institute of Technology, Cambridge, MA, USA

Itay Tsabary (3)
Technion, Haifa, Israel; IC3

Gerry Tsoukalas (9)
Information Systems Department, Questrom School of Business, Boston University, MA, USA

Catherine Tucker (12)
MIT Sloan, MIT, Cambridge, MA, USA

Maarten R.C. van Oordt (7)
Vrije Universiteit, Amsterdam, The Netherlands; Tinbergen Institute, Amsterdam, The Netherlands

Ganesh Viswanath-Natraj (11)
Warwick Business School, University of Warwick, Coventry, UK

Yang Yang (10)
Tsinghua University Institute of Economics, Beijing, China

Jiangshan Yu (2)
Monash University, Melbourne, Australia

Yu Zhu (7)
Bank of Canada, Ottawa, ON, Canada

Distributed Computing Meets Game Theory

Fault Tolerance and Implementation with Cheap Talk

Joseph Y. Halpern
Cornell University, Ithaca, NY, USA

Abstract

Traditionally, work in distributed computing has divided the agents into “good guys” and “bad guys”. The good guys follow the protocol; the bad guys do everything in their power to make sure it does not work. By way of contrast, game theory has focused on “rational” agents, who try to maximize their utilities. Here I try to combine these viewpoints. Specifically, following the work of Abraham et al. [2], I consider (k, t)-*robust* protocols/strategies, which tolerate coalitions of rational players of size up to k and up to t malicious players. I focus in particular on the problem that economists have called implementing a mediator. That is, can the players in the system, just talking among themselves (using what economists call “cheap talk”) simulate the effects of the mediator (see, e.g., [3, 4, 5, 6, 8, 10, 11]). In computer science, this essentially amounts to multiparty computation [7, 9, 12]. Ideas from cryptography and distributed computing allow us to prove results on how many agents are required to implement a (k, t)-robust mediator just using cheap talk. These results subsume (and, in some cases, correct) results from the game theory literature.

The results of Abraham et al. [2] were proved for what are called *synchronous systems* in the distributed computing community; this is also the case for all the results in the economics literature cited above. In synchronous systems, communication proceeds in atomic rounds, and all messages sent during round r are received by round $r+1$. But many systems in the real world are *asynchronous*. In an asynchronous setting, there are no rounds; messages sent by the players may take arbitrarily long to get to their recipients. Markets and the internet are best viewed as asynchronous. Blockchain implementations assume *partial synchrony*, where there is an upper bound on how long messages take to arrive. The partial synchronous setting already shows some of the difficulty of moving away from synchrony: An agent i can wait to take its action until it receives a message from j (on which its action can depend). This cannot happen in a synchronous setting. Abraham, Dolev, Geffner, abnd Halpern [1] extend the results on implementing mediators to the asynchronous setting.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory and mechanism design

Keywords and phrases robust equilibrium, implementing mediators, asynchronous systems

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2021.1

Category Invited Talk

Related Versions The talk described by this paper is based on two papers: “Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation”, by Ittai Abraham, Danny Dolev, Rica Gonen, and Joseph Y. Halpern, which appeared in the *Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing*, pp. 53–62, 2006, and “Implementing mediators with asynchronous cheap talk”, by Ittai Abraham, Danny Dolev, Ivan Geffner, Joseph Y. Halpern, which appeared in *Proceedings of the 38th Annual ACM Symposium on Principles of Distributed Computing*, pp. 501–510, 2019.
Full Version: `https://arxiv.org/abs/1806.01214`

Funding Halpern’s work was supported in part by NSF grants IIS-1703846 and IIS-0911036, ARO grant W911NF-17-1-0592, MURI (MultiUniversity Research Initiative) under grant W911NF-19-1-0217, and a grant from Open Philanthropy.

References

1 I. Abraham, D. Dolev, I. Geffner, and J. Y. Halpern. Implementing mediators with asynchronous cheap talk. In *Proc. 38th ACM Symposium on Principles of Distributed Computing*, pages 501–510, 2019.

2 I. Abraham, D. Dolev, R. Gonen, and J. Y. Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In *Proc. 25th ACM Symposium on Principles of Distributed Computing*, pages 53–62, 2006.

3 I. Barany. Fair distribution protocols or how the players replace fortune. *Mathematics of Operations Research*, 17(2):327–340, 1992.

4 E. Ben-Porath. Cheap talk in games with incomplete information. *Journal of Economic Theory*, 108(1):45–71, 2003.

5 F. Forges. Universal mechanisms. *Econometrica*, 58(6):1341–64, 1990.

6 D. Gerardi. Unmediated communication in games with complete and incomplete information. *Journal of Economic Theory*, 114:104–131, 2004.

7 O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. 19th ACM Symposium on Theory of Computing*, pages 218–229, 1987.

8 Y. Heller. A minority-proof cheap-talk protocol. Unpublished manuscript, 2005.

9 A. Shamir, R. L. Rivest, and L. Adelman. Mental poker. In D. A. Klarner, editor, *The Mathematical Gardner*, pages 37–43. Prindle, Weber, and Schmidt, Boston, MA, 1981.

10 A. Urbano and J. E. Vila. Computational complexity and communication: coordination in two-player games. *Econometrica*, 70(5):1893–1927, 2002.

11 A. Urbano and J. E. Vila. Computationally restricted unmediated talk under incomplete information. *Economic Theory*, 23(2):283–320, 2004.

12 A. Yao. Protocols for secure computation (extended abstract). In *Proc. 23rd IEEE Symp. Foundations of Computer Science*, pages 160–164, 1982.

General Congestion Attack on HTLC-Based Payment Channel Networks

Zhichun Lu
Cryptape, Hangzhou, China

Runchao Han
Monash University, Melbourne, Australia
CSIRO-Data61, Melbourne, Australia

Jiangshan Yu
Monash University, Melbourne, Australia

Abstract

Payment Channel Networks (PCNs) have been a promising approach to scale blockchains. However, PCNs have limited liquidity: large-amount or multi-hop payments may fail. The major threat of PCNs liquidity is *payment griefing*, where the adversary who acts as the payee keeps withholding the payment, so that coins involved in the payment cannot be used for routing other payments before the payment expires. Payment griefing gives adversaries a chance to launch the *congestion attack*, where the adversary griefs a large number of payments and paralyses the entire PCN. Understanding congestion attacks, including their strategies and impact, is crucial for designing PCNs with better liquidity guarantees. However, existing research has only focused on the specific attacking strategies and specific aspects of their impact on PCNs.

We fill this gap by studying the *general congestion attack*. Compared to existing attack strategies, in our framework each step serves an orthogonal purpose and is customisable, allowing the adversary to focus on different aspects of the liquidity. To evaluate the attack's impact, we propose a generic method of quantifying PCNs' liquidity and effectiveness of the congestion attacks. We evaluate our general congestion attacks on Bitcoin's Lightning Network, and show that with direct channels to 1.5% richest nodes, and $\sim$ 0.0096 BTC of cost, the adversary can launch a congestion attack that locks 47% ($\sim$280 BTC) coins in the network; reduces success rate of payments by 16.0%$\sim$60.0%; increases fee of payments by 4.5%$\sim$16.0%; increases average attempts of payments by 42.0%$\sim$115.3%; and increase the number of bankruptcy nodes (i.e., nodes with insufficient balance for making normal-size payments) by 26.6%$\sim$109.4%, where the amounts of payments range from 0.001 to 0.019 BTC.

2012 ACM Subject Classification Security and privacy → Distributed systems security

Keywords and phrases Blockchain, PCN, Congestion

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2021.2

Related Version *Full Version*: `https://eprint.iacr.org/2020/456`

1 Introduction

Public blockchains suffer from limited throughput. Payment Channel Network (PCN) – introduced by the Lightning Network (LN) [17] – is one of the promising ways to scale blockchains. Payment channels enable *off-chain* payments, i.e. payments that do not need to be recorded on the blockchain. To open a payment channel, two nodes collateralise some coins in a joint address. They can make a payment by signing a new transaction that updates their balances. To close the channel, one of the two nodes commits the transaction recording the latest balance allocation to the blockchain. If two nodes do not have a direct channel, they can make payments to each other using *multi-hop payments*, i.e., payments going through one or more intermediate channels. In a multi-hop payment, the payer has to find a path that directs him to the payee. The payment is made by updating balances of these channels in an atomic way. The atomic update can be achieved by Hash Time Locked Contracts (HTLCs): the payment in each hop is locked by a hash value chosen by the payee, and all payments proceed if the payee reveal a hash value's preimage before a timeout to redeem the payment from the payer, otherwise the payment will expire. In a HTLC-based multi-hop payment, the payee chooses a preimage, and nodes make HTLC payments on all involved channels with this preimage's hash value. Revealing this preimage activates these HTLC payments simultaneously.

A well-known attack on HTLC-based multi-hop payments is *payment griefing* [19], where the adversary makes a payment and withholds the preimage, so that coins involved in this payment are locked and cannot be used in other payments before the griefing payment expires. Thus, payment griefing can reduce the PCN's liquidity, i.e. the ability of routing payments. In addition, payment griefing is free, as the payer does not need to pay anything for failed payments. Moreover, payment griefing is also unaccountable, as 1) the victim cannot distinguish between a normal failed payment and a griefing payment, and 2) the intermediate nodes can not know the payer and payee's identity.

Griefing opens an important attack vector on HTLC-based PCN's liquidity, namely the *congestion attack* [7, 14]. In a congestion attack, the adversary initiates a large number of concurrent payments and griefs them. Consequently, some channels hit the limit of *max_concurrent_htlcs*, i.e., the number of concurrent unsettled payments allowed in the channel, and therefore cannot route payments before the adversary's payments expire. By launching a large-scale congestion attack, the entire PCN can be paralysed, i.e., the PCN cannot route further payments.

Understanding congestion attacks is important for understanding PCNs' liquidity and therefore future PCN design. However, congestion attacks are still a new concept and haven't been well-studied yet. While existing research [7, 14] only considers *max_concurrent_htlcs* as an exhaustible resource, it's unclear whether there exists other resources that can be exhausted to create congestion. In addition, existing congestion attacks apply a rather straightforward attack strategy, which will be analysed in detail in §7. Moreover, we also observe that liquidity – the congestion attack's target – is not well-defined yet. Besides the amount of locked balance and the number of locked channels mentioned in Mizrahi et al. [14], some other metrics such as success rate of payments, fee of payments, and number of attempts for making a payment have direct indications on the PCN's liquidity. Congestion attacks over these metrics are not explored before.

In this paper, we fill this gap by introducing *general congestion attack*, which generalises the existing congestion attack in terms of attack strategies and targeted metrics. We introduce a framework for launching congestion attacks, where the adversary generates Sybil nodes connecting to a carefully chosen set of nodes, establishes channels with them, initiates numerous multi-hop payments between its nodes, and griefs these payments simultaneously. Compared to existing studies that put less effort on the order of payments to be griefed [14, 22], we provide five strategies for ranking these payments, and each strategy focuses on some specific aspects of liquidity. To quantify the effectiveness of congestion attacks, we introduce a generic method of quantifying PCNs' liquidity. We evaluate the congestion attack on Bitcoin's LN – the first and most well-known PCN. Our results show that congestion attacks can significantly damage the liquidity of PCNs. In particular, with direct channels to 1.5% richest nodes, the adversary can launch a congestion attack that locks 47% ($\sim$280 BTC) coins in the network; reduces success rate of payments by 16.0%$\sim$60.0%; increases fee of payments by 4.5%∼16.0%; increases average attempts of payments by 42.0%∼115.3%; and increase the number of bankruptcy nodes by 26.6%∼109.4%, where the amounts of payments range from 0.001 to 0.019 BTC.

While being effective, our general congestion attacks are cheap to launch. The only cost of general congestion attacks is the fee for establishing channels. Our evaluation shows that, a successful attack on LN requires channel fee of approximately 0.0096 BTC. The adversary does not lose its custody (i.e., coins in the channel) during the attack, as payments for griefing will expire.

Section 2 provides the background of PCNs and griefing. Section 3 describes the security model and the congestion attack. Section 4 describes the method of quantifying PCNs' liquidity. Section 5 evaluates congestion attacks on LN. Section 6 discusses the cost of congestion and strategy to utilise it for making a profit. Section 7 reviews relevant literature, and provides a quantitative comparison between the general congestion attack and the existing ones. Section 8 concludes this paper.

2 Background

2.1 Payment Channel Networks

Lightning Network (LN) [17] introduces the idea of Payment Channel Networks. A payment channel allows two parties to pay each other without the need to publish every payment to the blockchain. Instead, they collateralise their coins into a 2-of-2 multi-signature address. They can make payments by mutually signing new transactions with updated balances. They can make payments with each other by mutually signing new transactions with updated amounts of their collateralised coins. To close the channel, one party commits the latest state of channel balance to the blockchain, and coins in the channel will be allocated to both parties accordingly.

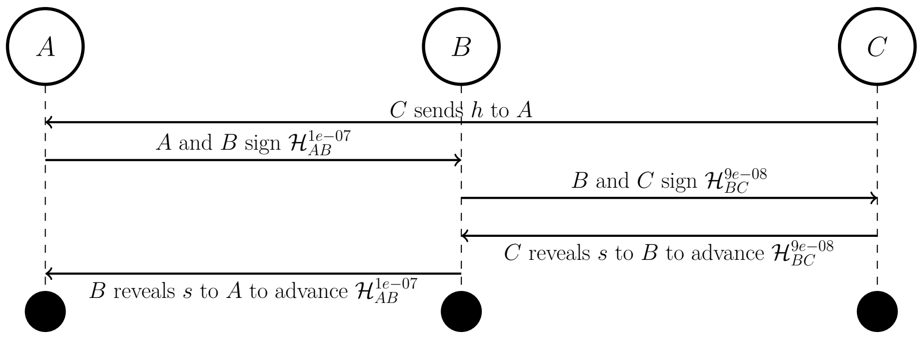

A
B
C
C sends h to A
A and B sign $\mathcal{H}_{AB}^{1e-07}$
B and C sign $\mathcal{H}_{BC}^{9e-08}$
C reveals s to B to advance $\mathcal{H}_{BC}^{9e-08}$
B reveals s to A to advance $\mathcal{H}_{AB}^{1e-07}$

Figure 1 A multi-hop payment from A to C via an intermediate node B.

The system can be further extended to support multi-hop payments. Most multi-hop payment protocols are based on Hash Time Locked Contracts (HTLCs). HTLC is a contract between two parties which guarantees that a payment will be made if the payee shows the preimage of a hash value before a negotiated block height on the blockchain. If the payee does not show the preimage and the timeout expires, the payment is deemed invalid.

Figure 1 describes a multi-hop payment where A pays 9e-08 BTC to C via an intermediate node B in Bitcoin's LN. First, C chooses a random string s as preimage and sends its hash value $h = H(s)$ to A, where $H(\cdot)$ is a cryptographic hash function. A then signs a HTLC contract $\mathcal{H}_{AB}^{1e-07}$ with B stating "A will pay 1e-07 BTC to B if B can show the value of s

within (e.g.) 144 blocks". B also signs a HTLC contract $\mathcal{H}_{BC}^{9e-08}$ with C saying that "B will pay 9e-08 BTC to C if C can show the value of s within (e.g.) 138 blocks". Then C shows s to B to redeem 9e-08 BTC in $\mathcal{H}_{BC}^{9e-08}$ from B. Meanwhile, B can redeem 1e-07 BTC in $\mathcal{H}_{AB}^{1e-07}$ from A by revealing s to A. B is incentivised to reveal s, as B does not want to lose money. The timelock of AB is set to be longer than BC, so B always has sufficient time to reveal s to A.

By routing this payment, B gets 1e-08 BTC from A. This is known as "fee", which is paid by the payer and is used for encouraging nodes to route multi-hop payments. In LN, fee consists of a fixed base fee and proportional fee that fluctuates according to the congestion level of the network. To minimise the cost, payers usually search for a path with the least fee when making payments.

2.2 Payment Griefing

If the payee C reveals the preimage on time and the intermediate node B is rational, the multi-hop payment will happen. However, as we mentioned before, there exists an attack called *payment griefing* [11], where the payee withholds the preimage until HTLCs expire. Before HTLC expires, coins involved in all channels of this payment are locked and cannot route other payments.

Payment griefing is a threat to PCNs' liquidity. If a big portion of coins in a PCN are locked, the PCN will no longer be able to route payments. Payment griefing is cheap, as the payment does not really happen and the payer does not pay for the fee to intermediate nodes. Identifying payment griefing can be hard, as nodes cannot distinguish whether the withholding is due to network delay, on purpose, or by accident. If the PCN's routing protocol is privacy-preserving, payment griefing can even be launched anonymously. For example, Bitcoin's LN adopts Sphinx [5], where each intermediate node only has the knowledge of nodes who directly connect with him.

2.3 Congestion attack

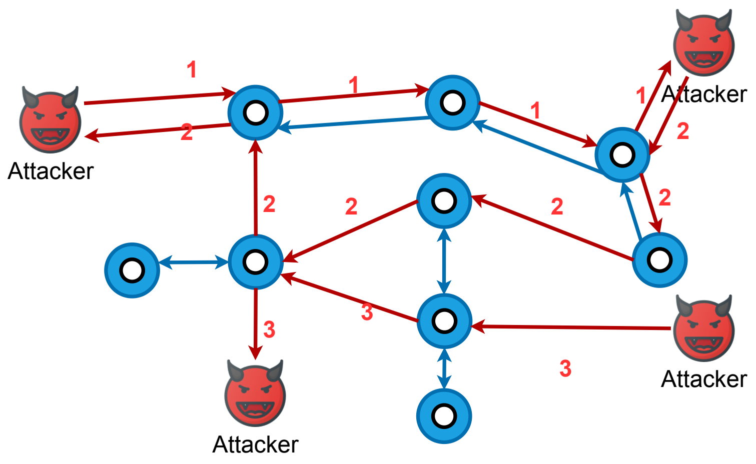


Figure 2 Congestion attack.

In a congestion attack, the adversary establishes payment channels with existing nodes in the PCN, and make numerous multi-hop payments between its nodes simultaneously. Then, the adversary withholds preimages until these payments expire. Before that, coins locked in these payments cannot be used in other payments. If the adversary has sufficient custody, it can lock a great portion of coins in the PCN so that the PCN may be paralysed. Figure 2 shows the intuition of the congestion attack, where the adversary generates Sybil nodes connecting to a carefully chosen set of nodes, establishes channels with them, initiates numerous multi-hop payments between its nodes, and griefs these payments simultaneously.

To reduce the custody required for an attack, Mizrahi et al. [14] proposed to lock the channel by initiating numerous payments with small amounts to occupy all *max_concurrent_htlcs*, the maximum number of concurrent payments in a channel. In addition, they propose three strategies for enumerating payment paths. Tikhomirov et al. [22] provided another strategy, where the adversary attacks a single channel rather than a path for each step.

We compare the strategies of our attack with those of Mizrahi et al. [14] and Tikhomirov et al. [22] in §3.2, and compare the cost and effectiveness in §7. The result show that. If the attacker is fee-sensitive, then our attack is preferred because our fees are 16% of and 5% of other two. Whereas, if the attacker has a restricted custody in hand, then the attack by person Mizrahi et al. is more preferred, as the custody required is only 1.5% of our attack (in case of locking 41% network's capacity).

3 General congestion attack

3.1 Model

We consider a HTLC-based PCN that is identical to Bitcoin's Lightning Network as described in §2.1. We model the PCN as a weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ is the set of nodes in the network, and all v_* in the remainder of the paper refer to a specific element (i.e. a node) in $\mathcal{V}$. $\mathcal{E}$ is a set of tuples $(v_s, v_t, \mathit{capacity})$, which represents a channel with a capacity of *capacity* from v_s to v_t. Since channels in LN is bi-directional, a channel is represented in the graph as two opposite edges $(v_s, v_t, \mathit{capacity})$ and $(v_t, v_s, \mathit{capacity})$. A payment P is a dictionary $\{\mathit{amount} : \alpha, \mathit{path} : \mathit{path}\}$, where α is the amount of coins that the payer wants to pay to the payee and *path* is the path of payment (a list of edges). Then, $\mathcal{P}$ denotes list of payments enumerated by the attacker. Meanwhile, we use l to refer the length of *path*. For simplicity, we do not consider the impact of timelocks on our attack. The payer also needs to pay some fee f_i for each intermediate node i on the path. We can now get the size of a multi-hop payment θ as

$$\theta = \alpha \cdot l + \sum_{i=2}^{l} f_i$$

In addiciton, we use Γ_v to refer the total capacity of all channels connecting to v. Therefore, the richest node we defined earlier is the node with the largest Γ_v.

We consider nodes in the PCN are rational. Each honest node in a multi-hop payment will reveal the preimage of the hashlock to the upstream node once the node knows it. We assume a malicious adversary, who has sufficient coins and aims at paralysing the entire PCN with minimal cost. The adversary does not control any node in the beginning, but has the knowledge of the network topology and each channel's capacity and fee policy. The information can be retrieved from PCN's P2P protocol, evidenced by existing studies [3, 9].

When the adversary establishes a channel with a node, the node is willing to provide sufficient capacity. According to liquidity providers such as Bitrefill [2], purchasing capacity from nodes is easy and costs negligible coins. Moreover, if an adversary just wants to attack PCN for a period of time, it can use the channel lease marketplace like lightning pool [15] to get incoming liquidity at a much lower cost.

3.2 Attack framework

We generalise the congestion attack in terms of attack strategies and liquidity metrics. We propose a framework for launching congestion attacks. The framework consists of four steps as follows.

1. **Node selection:** the adversary chooses a set of nodes and establishes channels with them.
2. **Payment enumeration:** the adversary enumerates all payments between its nodes.
3. **Path ranking:** the adversary orders these payments.
4. **Launching attack:** the adversary starts to make and abort payments in this order.

While existing congestion attacks [14, 22] start by enumerating griefing paths or griefing channels, our attack chooses nodes in the beginning and enumerates paths within the given set of nodes. Such design allows us to divide the attack into multiple steps with orthogonal purposes, revealing the complete design space for congestion attacks. Specifically, the adversary decides resources (e.g., channel capacity or the max_concurrent_htlc parameter) to congest when enumerating payments, and decides its focus of liquidity metrics when ranking payments.

In addition, in existing congestion attacks [14, 22], the adversary has to create new channels when attacking a new channel or path. Therefore, the adversary has to establish a large number of channels, which incurs additional transaction fees on the underlying blockchain. In contrast, our attack chooses nodes in the beginning and enumerates paths within the given set of nodes, and therefore requires much fewer channels.

3.3 Node selection

The adversary's first step is to join the PCN by establishing channels with existing nodes. We analyse the adversary's strategy on the type of nodes and the number of nodes to establish channels with.

We suggest establishing channels with the richest (w.r.t. total capacity of involved channels) nodes (which we call *hubs*) in the network, as they are likely to route more griefing payments than a normal node. If establishing channels with nodes with little capacity, the adversary has to establish channels with more nodes, leading to more fee on establishing channels.

The number of nodes to establish channels with depends on how the adversary enumerates griefing payments (i.e., the second step). By establishing channels with sufficient nodes, the total size of enumerated payments will take the majority of the network capacity, and therefore the congestion attack will take effect. Later in §5.1, we will show that for Bitcoin's Lightning Network, by establishing channels with the top 1.5% richest nodes the enumerated payments can occupy 81% of the network capacity ideally (47% in the experiment).

3.4 Enumerating payments

After establishing payment channels, the adversary enumerates all possible payments between its nodes. To this end, the adversary has to find all paths between each pair of its nodes, and calculate the maximum amount that each path can afford. Our payment enumerating algorithm builds upon the Ford–Fulkerson algorithm [8] - a maximum flow algorithm in graph theory. Maximum flow is a classic problem in graph theory, which aims at finding the maximum amount of flow that the network allows from a source to a sink. Ford–Fulkerson

Algorithm 1 Enumerating payments.

```
Input:
 1: The entire network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
 2: The adversary's node list $\mathcal{N}$
Output:
 3: The list of payments $\mathcal{P}$
 4: $\mathcal{P} \leftarrow []$
 5: for $(v_1, v_2)$ in $\mathcal{N}.combination()$ do
 6:     $path\_list \leftarrow \mathsf{BFS}(\mathcal{G}, v_1, v_2)$
 7:     for $path \in path\_list$ do
 8:         $P \leftarrow \{path : [], amount : 0\}$
 9:         $P.path \leftarrow path$
10:         $capacity\_list \leftarrow$ [edge.capacity for edge in path]
11:         $\alpha \leftarrow min(capacity\_list)$                  ▷ Max viable amount
12:         if $\alpha = 0$ then continue                      ▷ This path is not viable
13:         $P.amount \leftarrow \alpha$                          ▷ $P$ is a viable payment
14:         Append $P$ to $\mathcal{P}$
15:         Consume $P$ in $\mathcal{G}$
16:     end for
17: end for
18: return $\mathcal{P}$
```

algorithm is one of the most effective algorithms to solve the maximum flow problem. Given a weighted directed graph and two vertices. The Ford-Fulkerson algorithm uses Breadth-First Search (BFS) [12] to find all paths between these two vertices. For each path, the maximum viable amount is the minimum weight of edges. Algorithm 1 describes the process of enumerating payments in Python syntax. First, it enumerates the binary combinations of nodes who the adversary establishes channels as the starting and ending points of the path. Second, it executes BFS to find all paths between each two adversary nodes similar Ford–Fulkerson. Third, we derive the most viable amount using the least channel capacity for each path. Last, it consumes this payment from the graph and adds this payment to our payment list, i.e., we subtract the amount of this payment from the capacity of all channels on the path.

3.5 Ranking payments

Different griefing payments have different impacts on the PCNs' liquidity. With limited balance, the adversary has to start from griefing important payments for maximising the attack's effect.

We first consider three ranking criteria, namely the length, amount, and size θ of a payment. *Rank-by-length* aims at maximising the effect while minimising the cost, as long payments lock most capacity with the least amount. *Rank-by-amount* aims at attacking the channel with large capacity. Since real-time balance in LN cannot be seen, payer tends to prefer to go through channels with large capacities to reduce the number of attempts. *Rank-by-size* aims at maximising the attack effect without considering the custody, as payments with large sizes cost most collateral.

Inspired by B'eres et al. [3], we consider *rank-by-fee*, where the adversary starts from attacking channels with lower fees. This aims at maximising the average channel fee of normal payments after the attack. The ranking criterium $Score(P)$ for payment P is calculated as

$$Score(P) = -\frac{\sum_{i=2}^{l} f_i}{l}$$

In addition, we consider the bankruptcy rates presented by Dandekar et al. [4,18,21] as a ranking criterium. The adversary first attacks channels that make most nodes "bankrupt". Dandekar et al. introduced credit networks [4], where nodes are connected by edges with a limited resource called *credit*. We model the PCN as a credit network following Ramseyer et al. [18], where each channel's credit is its capacity. Liquidity in a credit network is quantified as the probability that nodes become *bankrupt*, i.e., loss of all credit. Dandekar et al. [4] proved that the probability that a node v goes bankrupt is upper-bounded by $\frac{1}{\Gamma_v+1}$, where Γ_v is the total capacity of all channels connecting to v. Griefing can be seen as "removing" the capacity of nodes, and therefore increases the probability of nodes becoming bankrupt. In *rank-by-bankrupt*, the adversary first attacks payments that reduce the most mathematical expectations of the probability of nodes becoming bankrupt.

The bankruptcy criterium is quantified as the total increased probability $Score(P)$ of nodes in P becoming bankruptcy

$$Score(P) = Score_t(v_1, \alpha) + \sum_{i=1}^{l} Score_i(v_i, \alpha) + Score_t(v_{l+1}, \alpha)$$

where $Score_t(v_1, \alpha)$ and $Score_t(v_{l+1}, \alpha)$ are the increased probability of node v_1 and v_{l+1}, respectively, and $Score_i(v_i, \alpha)$ is the increased probability of intermediate nodes v_i where $i \in [2, l]$. For node $v = v_1$ or v_{l+1}, the capacity is reduced by α, so

$$Score_t(v, \alpha) = \frac{1}{\Gamma_v - \alpha + 1} - \frac{1}{\Gamma_v + 1}$$

Meanwhile, for intermediate nodes $v = v_i$ where $i \in [1, l-1]$, the capacity is reduced by 2α, so

$$Score_i(v, \alpha) = \frac{1}{\Gamma_v - 2\alpha + 1} - \frac{1}{\Gamma_v + 1}$$

3.6 Launching attack

To obtain a list of griefing payments before launching the attack, we use channels' capacities rather than their balances for determining the amounts of payments. As balances are fluctuating in real-time, some of the enumerated payments may not succeed during the attack. In real-world PCNs, if a node cannot route a payment, the node will reply to the payer with an error message, e.g., LN calls this error *InsufficientFunds*. Thus, we introduce a retry mechanism that, when a griefing payment is rejected, the adversary reduces its amount by a parameter *step*, and retries the same path until it is successful or the amount reaches zero. Algorithm 2 describes the attack process. To avoid being detected due to the retry pattern, the adversary can obfuscate the payment pattern, e.g., by dividing payments into multiple ones with random amounts.

Algorithm 2 Launching attack.

```
Input:
1:  The list of ranked payments $\mathcal{P}_{ranked}$
2:  The dropping step ratio $step\_ratio$
3:  The custody of the adversary $B$
4:  for $P$ in $\mathcal{P}_{ranked}$ do
5:      if $B \leq 0$ then
6:          return ;
7:      end if
8:      $step\_amount \leftarrow step\_ratio * P.amount$
9:      while $True$ do
10:         $response \leftarrow make\_payment(P)$
11:         if $response =$ InsufficientFunds then
12:             $P.amount = P.amount - step\_amount$
13:             if $P.amount \leq 0$ then break
14:             continue
15:         end if
16:         $B = B - P.amount$
17:         break
18:     end while
19: end for
```

4 Quantifying PCNs' liquidity and congestion attacks' impact

We propose a generic method of quantifying PCNs' liquidity and congestion attacks' impact. In our method, we generate a batch of payments, simulate them on the PCN, and calculate liquidity metrics. The liquidity metrics include the success rate, the average cost and the number of attempts of payments, and the number of bankruptcy nodes. A congestion attack's impact is quantified as the liquidity difference before and after the attack.

4.1 Generating payments for simulation

We follow the approach of Béres et al. [3] to test PCNs' liquidity. Specifically, we generate a batch of n payments, of which payers and payees are random and the amount x_t is fixed. We test multiple batches of payments with different amounts to cover regular payment scenarios, which will be discussed in §5.

We simulate these payments in the PCN. Each payment is allowed to try r times to find a viable path. If it finds a path within r tries, we consider it successful, otherwise failed. Then the payments can be categorized into three states according to the status before and after the attack, namely *added*, *survived*, or *removed*. *Added* means the payment fails before the attack but is successful after the attack. *Survived* means the payment is successful both before and after the attack. *Removed* means the payment is successful before the attack but fails after the attack. Since our analysis is based on successful payments, payments that fail both before and after the attack are ignored here.

Interestingly, some payments are *added*. Assuming a payments $P_1 = A \rightarrow B \rightarrow C \rightarrow D$ fails before the attack since BC has insufficient balance. For example, if some successful payments that would have gone through BC failed after the attack, then channel BC would become available to P_1. Another example is that the attack may cause some payments to change their paths. Suppose a successful payment P_2 originally went through EF and was forced to go through CB after the attack, then the balance of BC will increase and make it have enough balance to route $P1$.

4.2 Calculating liquidity metrics

We derive the PCNs' liquidity from the execution results of the simulated payments. We consider the following five metrics: 1) amount of locked funds, 2) success rate of payments, 3) fee of payments, 4) average attempt times of payments and 5) the number of bankruptcy nodes. A congestion attack's impact is quantified by the difference of liquidity metrics before and after the attack.

5 Evaluation of congestion attacks

In this section, we evaluate congestion attacks on Bitcoin's Lightning Network (LN), the first and most well-known PCN. We analyse the impact of congestion attacks with different strategies in terms of the defined five liquidity metrics. Our results show that the adversary who adopts congestion can severely limit the functionality of the entire PCN. Specifically, the adversary can launch a congestion attack that locks 47% (∼280 BTC) coins in the network; reduces success rate of payments by 16.0%∼60.0%; increases fee of payments by 4.5%∼16.0%; increases average attempts of payments by 42.0%∼115.3%; and increase the number of bankruptcy nodes by 26.6%∼109.4%, where the amounts of payments range from 0.001 to 0.019 BTC.

5.1 Experimental setting

We simulate and implement our attack using Python 3.7.4 and NetworkX. For simplicity, we implement all algorithms sequentially. Adversaries can use multi-threaded programming to speed up the algorithm if they prefer efficiency. The topology provided by B'eres et al. [3] is the snapshot of the LN in 2019 (we checked the network snapshot for 2021 and found the topology to be similar to 2019, so we believe the results are similar of simulation on the 2021 snapshot). The snapshot also includes the fee policy for each channel as well as the capacity. Since the balance distribution characteristics of LN are not publicly available, we apply the random uniform distribution for initialising the channels' balances similar to existing studies [3]. To amortise the bias from randomness, we run each group of simulations with a certain strategy and custody level for ten times. Our results show that the coefficient of variation for the quantitative impact of the different balance distributions is only 1%.

In the real-world scenario, payments may sometimes fail, as nodes cannot know the real-time balances of channels they do not involve. LN introduces a *success probability* mechanism to optimise the routing. Specifically, if intermediate node A is unable to forward a payment because of insufficient balance, then it will return an error to the sender. The sender will temporarily reduce the success probability of this node. The path finding mechanism of LN is finding the shortest path on a weighted graph. For simplicity, we set the weight as channel fee. The routing algorithm is the plain Dijkstra [6] algorithm. When an attempt fails, we temporarily remove the first node on the current path with insufficient balance and try again. A payment is allowed to try r times for finding a viable path. If it finds a path within r tries, it is successful, otherwise we consider it fails.

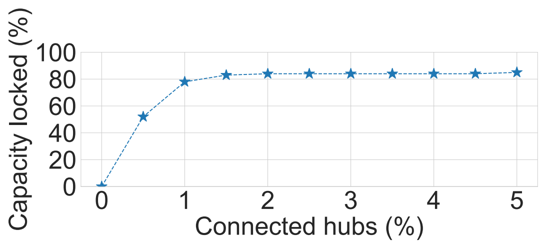
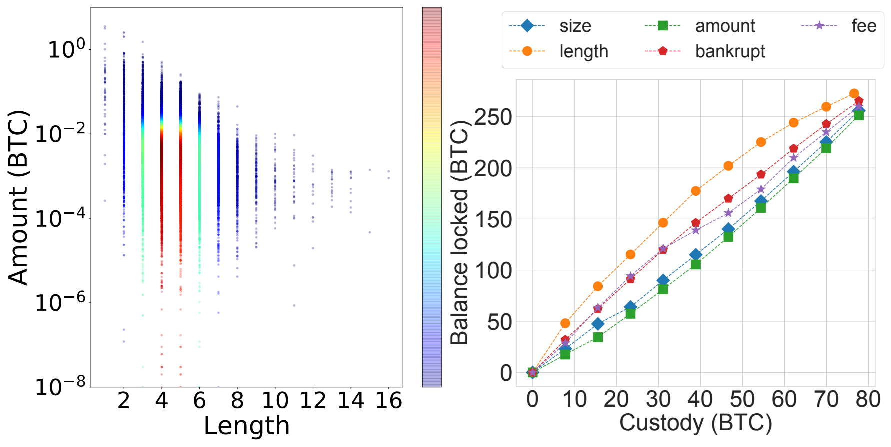


Figure 3 The percentage of connected hubs v.s. locked capacity.



(a) Distribution of enumerated payments. **(b)** Locked balance.

Figure 4 Characterisation of enumerated payments and the amount of locked balance.

We test attacks with different levels of custody of the adversary, i.e., $\{7.7, 15.4, \ldots, 77\}$ BTC, all ranking criteria in Section 3.5, and $step\_ratio = 0.1$ in Algorithm 2. When testing LN's liquidity, we pick batch size $n = 7000$ for testing liquidity(which is identical to to existing works [3]), payment amount $x_t \in \{0.001, 0.007, 0.013, 0.019\}$ BTC, and payment retry times $r = 10$. In total, we ran $10 * 4 * 10 * 5 = 2000$ (retry times * # of payment amounts * # of different custody levels * # of strategies) simulations. We consider the threshold of bankruptcy as 0.006 BTC, which is the average amount of payments in LN [3].

We test the percentage of the capacity that the adversary can lock by establishing channels with different numbers of richest nodes in LN. Figure 3 shows that, by establishing channels with the top 1.5% (42) richest nodes, the enumerated payments take $\sim$ 83% of the capacity of the entire network. In addition, the total amount of enumerated payments converges with the percentage of hubs increasing.

5.2 Impact of congestion attacks

We simulate the general congestion attack with five strategies in §3.5, and evaluate their impact in terms of the five metrics defined in §4. Figure 4 summarises the results under rank-by-length strategy, figures of all strategies appear in the full version available online [13]. For Figure 4b and Figure 5, the baseline (when $x = 0$) is the scenario without any attack.

As mentioned before, we establish channels with the 42 richest nodes in the network. Algorithm 1 enumerates 35,402 payments in total. Figure 4a visualises the distribution of these payments w.r.t. their amounts and lengths. Red indicates there are many griefing

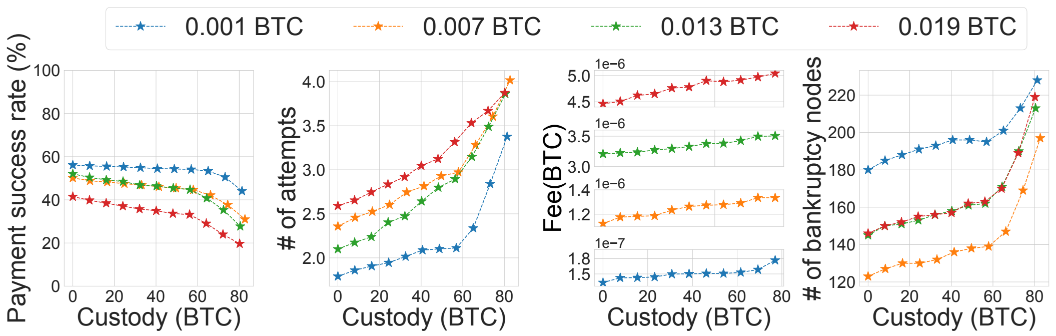


Figure 5 Overview of impacts of rank-by-fee.

paths under that path length and payment amount, while blue means the opposite. The amount ranges from zero to 0.1 BTC, while the length ranges from 1 to 13. On average, most payments have a length of 3 $\sim$ 6 and an amount of $1e-05 \sim 0.01$ BTC.

With a custody of 80 BTC (13% of the total capacity), an adversary can lock 280 BTC (47% of the total capacity) in LN, where rank-by-length is the most efficient strategy for locking balance. The average length of griefing payments is 3.8, which implies that there is room for optimisation of our path enumeration algorithm, since LN allows a maximum payment length of 20 hops.

Figure 5 shows the result of the rank-by-fee strategy on LN as an example. With the rank-by-fee strategy and 7 $\sim$ 80 BTC as custody, the attack can reduce the payments' success rates by 21.4%$\sim$52.3%, increase the fee by 9.3%$\sim$27%, increase the number of attempts by 50%$\sim$88.7%, and increase the number of bankrupt nodes by 26.7%$\sim$60%.

6 Discussion

The budget of launching congestion attacks is twofold: 1) *channel fee* for establishing channels and 2) *custody* deposited into channels. When preparing for a congestion attack, the adversary needs to pay the transaction fee for opening channels. Transaction fee is negligible as analysed in §5.1. After the congestion attack, the custody is refunded as payments are expired. For LN, the required custody is 77 BTC (13% of the network capacity). In Bitcoin, there are more than 10,000 addresses with more than 157 BTC [1], making them having sufficient capacity to launch a congestion attack.

The adversary can apply griefing on other nodes' channels, so that more payments go through its controlled channels. To receive most fees following this approach, the adversary redirects as many payments to its channels as possible. Existing research [23] shows that the probability is proportional to the adversarial node's *betweenness centrality*, while maximising the *betweenness centrality* by removing channels can be formalised as the *destructive betweenness improvement* problem that is NP-hard [10]. To our knowledge, there exists no approximation algorithm to solve this problem, and we consider designing such algorism as future work.

7 Related work and comparison

7.1 Attacks on PCNs

Congestion attack was informally discussed by Lightning Network community [7]. Mizrahi et al. [14] first systematically studied the congestion attack on PCNs. In their proposed attack, the adversary makes a large number of small payments, in order to make channels hit *max_concurrent_htlcs*, the maximum number of concurrent payments. Tikhomirov et al. [22] used the same idea to lock the balance of the channel, but they only grief a single channel at a time.

The two congestion attacks focus on a single attack liquidity metric, or put limitations on the attack strategy, and therefore can be seen as special cases of our general congestion attack. In addition, as their attacks focus on a single path or channel at a time, the adversary has to establish new channels when attacking a new path or channel. Establishing a large number of channels makes the adversary easier to be identified, and existing nodes may not be willing to establish too many channels in a short time period. Moreover, to occupy *max_concurrent_htlcs*, the adversary in their two attacks has to make a large number of concurrent payments compared to our attack. This also makes the adversary's behaviour easier to be identified.

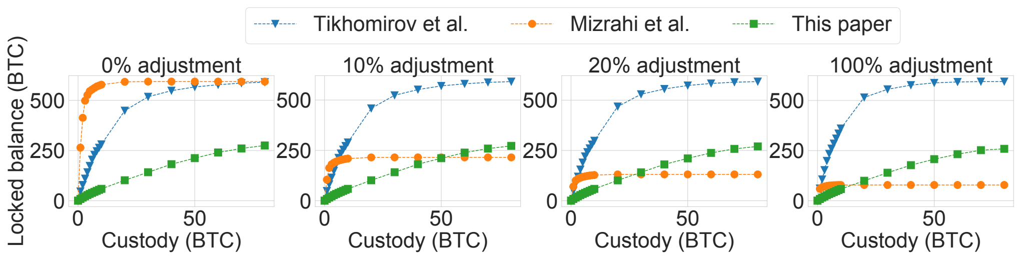
There have been attacks on PCNs with different goals. In the lockdown attack [16], the adversary griefs the victim's channels to isolate it from the network. In the hijacking attack [23], the adversary publishes channels with small fee to attract payments, and withhold all payments through its channels. Rohrer et al. [20] discussed two attacks, namely channel exhaustion and node isolation. While congestion attacks aim at paralysing the entire PCN, these three attacks aim at exhausting individual channels or isolating individual nodes.

7.2 Quantitative comparison with existing congestion attacks

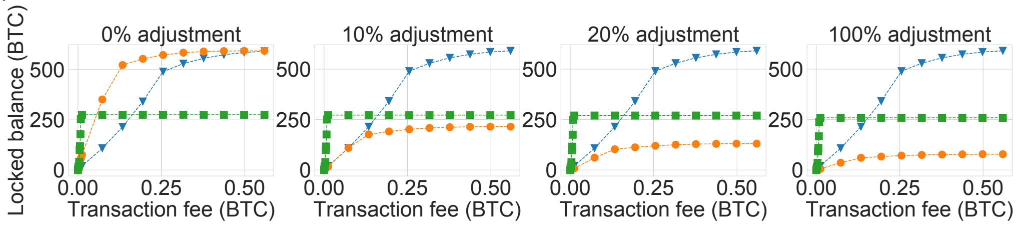
We quantitatively compare existing congestion attacks [14, 22] with ours w.r.t. different budget level of custody and channel fee and different *max_concurrent_htlcs* value distribution. For both attacks, we simulate the capacity-first strategy. The strategy iterates the following process: when a path is enumerated, calculate the total capacity of involved channels whose *max_concurrent_htlcs* values have been filled, then remove these channels from the network. Locking a channel by using *max_concurrent_htlcs* takes *max_concurrent_htlcs* * 2 payments (as a channel has two directions). The smallest payment amount is 5.46e-06 BTC (i.e. the dust limit). Thus, the custody required for griefing a path is 2 * *max_concurrent_htlcs* * 5.46e-06 BTC. When enumerating a path, we check whether both ends have channels with the adversary. If not, the adversary has to establish channels with them, leading to a fee of 0.0002 BTC ($\sim$ 18.89 USD at the time of writing).

To quantify the impact of *max_concurrent_htlcs*, we test the locked capacity when different portions of channels adjust *max_concurrent_htlcs*. Given the size limit of Bitcoin transactions, the maximum value of *max_concurrent_htlcs* in Bitcoin's LN is 483. Thus, we assume the adjusted value of *max_concurrent_htlcs* is uniformly distributed in interval $[1, 483]$. When the custody is limited, we assume the fee is unlimited, and vice versa.

Figure 6 shows the experimental results. Each experiment is repeated 10 times, and the variation of experimental results is about 2.4%. As the results are similar after the 20% channel adjustment, we skipped the simulation in 30%-90% for brevity. Figure 6a shows the performance of the three attacks under different custody. When all channels share the same *max_concurrent_htlcs*, Mizrahi et al.'s attack locks most capacity. When more



(a) Comparison of attacks with limited custody.



(b) Comparison of attacks with limited cost.

Figure 6 Comparison with congestion attack. x% adjustment means x% channels adjust their *max_concurrent_htlcs*.

channels adjust *max_concurrent_htlcs*, the locked capacity becomes less. This is because when channels in a path have different *max_concurrent_htlcs* values, the adversary can only congest the channel with the smallest *max_concurrent_htlcs* in this path by spamming this path only, making the strategy of Mizrahi et al. less effective. Meanwhile, Tikhomirov et al.'s attack and our attack are not affected by *max_concurrent_htlcs*. This is because our attack does not rely on *max_concurrent_htlcs* and the number of concurrent htlcs occupied by our attack averaged only 3.8 per channel, and Tikhomirov et al.'s attack focuses on a channel at a time. Figure 6b shows that, both Tikhomirov's and Mizrahi's attacks require more transaction fee compared to our attack. This is because, in their attacks, the adversary has to open a new channel when attacking a new path. With sufficient transaction fee, Tikhomirov's locks more money compared to our attack.

To lock in 250 BTC of liquidity. The attack by Mizrahi et al et al. requires 1 BTC of custody and pays a transaction fee of 0.05 BTC, the attack by person Tikhomirov et al. requires 8 BTC of custody and a fee of 0.15 BTC, while our attack requires 65 BTC of custody and a fee of 0.008 BTC. Therefore, if the attacker is fee-sensitive, then our attack is preferred because our fees are 16% of and 5% of other two. Whereas, if the attacker has a restricted custody in hand, then the attack by person Mizrahi et al. is more preferred, as the custody required is only 1.5% of our attack.

8 Conclusion

In this paper, we propose the general congestion attack on payment channel networks (PCNs). Our general congestion attack generalises the existing congestion attacks in terms of attack strategies, targeted metrics and optimisation techniques. We develop concrete steps for launching congestion attacks, and provide a generic method of quantifying PCNs' liquidity and effectiveness of congestion attacks. We evaluate our congestion attacks on Lightning Network – the first and most well-known PCN. Our evaluation results show that the congestion attack is cheap to launch and can greatly reduce the LN's liquidity.

References

1 BitInfoCharts. https://bitinfocharts.com/top-100-richest-bitcoin-addresses-1.html, 2020. [Online; accessed 20-September-2020].

2 Bitrefill. https://www.bitrefill.com/, 2020. [Online; accessed 20-September-2020].

3 Ferenc Béres, Istvan Andras Seres, and András A Benczúr. A cryptoeconomic traffic analysis of bitcoins lightning network. *arXiv preprint*, 2019. arXiv:1911.09432.

4 Pranav Dandekar, Ashish Goel, Ramesh Govindan, and Ian Post. Liquidity in credit networks: A little trust goes a long way. In *Proceedings of the 12th ACM conference on Electronic commerce*, pages 147–156, 2011.

5 George Danezis and Ian Goldberg. Sphinx: A compact and provably secure mix format. In *2009 30th IEEE Symposium on Security and Privacy*, pages 269–282. IEEE, 2009.

6 Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

7 EmelyanenkoK. Payment channel congestion via spam-attack. https://github.com/lightningnetwork/lightning-rfc/issues/182. Github, 2017.

8 Lester Randolph Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8:399–404, 1956.

9 Jordi Herrera-Joancomartí, Guillermo Navarro-Arribas, Alejandro Ranchal-Pedrosa, Cristina Pérez-Solà, and Joaquin Garcia-Alfaro. On the difficulty of hiding the balance of lightning network channels. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 602–612, 2019.

10 Clemens Hoffmann. Algorithms and complexity for centrality improvement in networks, 2017.

11 Akash Khosla, Evan Schwartz, and Adrian Hope-Bailie. Interledger rfcs, 0018 draft 3, connector risk mitigations. http://j.mp/2m2OvfP, Github, 2019.

12 Dexter C Kozen. Depth-first and breadth-first search. In *The design and analysis of algorithms*, pages 19–24. Springer, 1992.

13 Zhichun Lu, Runchao Han, and Jiangshan Yu. General congestion attack on htlc-based payment channel networks. *Cryptology ePrint Archive*, 2020.

14 Ayelet Mizrahi and Aviv Zohar. Congestion attacks in payment channel networks. *arXiv preprint*, 2020. arXiv:2002.06564.

15 Olaoluwa Osuntokun, Conner Fromknecht, Wilmer Paulino, Oliver Gugger, and Johan Halseth. Lightning pool: A non-custodial channel lease marketplace, 2020.

16 Cristina Pérez-Sola, Alejandro Ranchal-Pedrosa, J Herrera-Joancomartí, Guillermo Navarro-Arribas, and Joaquin Garcia-Alfaro. Lockdown: Balance availability attack against lightning network channels, 2019.

17 Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

18 Geoffrey Ramseyer, Ashish Goel, and David Mazières. Liquidity in credit networks with constrained agents. In *Proceedings of The Web Conference 2020*, pages 2099–2108, 2020.

19 Daniel Robinson. Htlcs considered harmful. In *Stanford Blockchain Conference*, 2019.

20 Elias Rohrer, Julian Malliaris, and Florian Tschorsch. Discharged payment channels: Quantifying the lightning network's resilience to topology-based attacks. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 347–356. IEEE, 2019.

21 Vibhaalakshmi Sivaraman, Weizhao Tang, Shaileshh Bojja Venkatakrishnan, Giulia Fanti, and Mohammad Alizadeh. The effect of network topology on credit network throughput. *arXiv preprint*, 2021. arXiv:2103.03288.

22 Sergei Tikhomirov, Pedro Moreno-Sanchez, and Matteo Maffei. A quantitative analysis of security, anonymity and scalability for the lightning network. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 387–396. IEEE, 2020.

23 Saar Tochner, Aviv Zohar, and Stefan Schmid. Route hijacking and dos in off-chain networks. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 228–240, 2020.

Tuning PoW with Hybrid Expenditure

Itay Tsabary
Technion, Haifa, Israel
IC3

Alexander Spiegelman
Novi Research, Herzliya, Israel

Ittay Eyal
Technion, Haifa, Israel
IC3

Abstract

Proof of Work (*PoW*) is a Sybil-deterrence security mechanism. It introduces an *external cost* to system participation by requiring computational effort to perform actions. However, since its inception, a central challenge was to tune this cost. Initial designs for deterring spam email and DoS attacks applied overhead equally to honest participants and attackers. Requiring too little effort does not deter attacks, whereas too much encumbers honest participation. This might be the reason it was never widely adopted.

Nakamoto overcame this trade-off in Bitcoin by distinguishing desired from malicious behavior and introducing *internal rewards* for the former. This mechanism gained popularity in securing permissionless cryptocurrencies, using virtual internally-minted tokens for rewards. However, in existing blockchain protocols the internal rewards directly compensate users for (almost) the same value of external expenses. Thus, as the token value soars, so does the PoW expenditure. Bitcoin PoW, for example, already expends as much electricity as Colombia or Switzerland. This amount of resource-guzzling is unsustainable, and hinders even wider adoption of these systems.

As such, a prominent alternative named *Proof of Stake* (*PoS*) replaces the expenditure requirement with token possession. However, PoS is shun by many cryptocurrency projects, as it is only secure under qualitatively-different assumptions, and the resultant systems are not permissionless.

In this work we present *Hybrid Expenditure Blockchain* (*HEB*), a novel PoW mechanism. *HEB* is a generalization of Nakamoto's protocol that enables tuning the external expenditure by introducing a complementary *internal-expenditure* mechanism. Thus, for the first time, HEB decouples external expenditure from the reward value.

We show a practical parameter choice by which *HEB* requires significantly less external consumption compare to Nakamoto's protocol, its resilience against rational attackers is similar, and it retains the decentralized and permissionless nature of the system. Taking the Bitcoin ecosystem as an example, *HEB* cuts the electricity consumption by half.

2012 ACM Subject Classification Security and privacy → Distributed systems security

Keywords and phrases Blockchain, Proof of work, Cryptocurrency, Environmental impact

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2021.3

Related Version *Extended Version*: `https://arxiv.org/abs/1911.04124` [63]

Funding This research was supported by the Israel Science Foundation (grant No. 1641/18), BSF, and the Technion Hiroshi Fujiwara cyber-security research center.

Acknowledgements We thank Or Sattath for his comments on preliminary versions of this work. Part of this work was performed when the authors worked at VMware Research, Israel.

Table 1 Security scheme rewards-expenses comparison.

Internal Rewards	**Expenses**		
	Negligible	*External*	*External & Internal*
Exist	PoS Blockchains [39, 29, 24, 30]	PoW Blockchains [50, 10]	*HEB* (this work)
Absent	Open systems (e.g., email)	Original PoW [19, 37]	

1 Introduction

Permissionless systems are susceptible to *Sybil attacks* [18] where a single attacker can masquerade as multiple entities. To mitigate such attacks, *Proof of Work* (*PoW*) [19, 36] security schemes introduce *external costs*, making attacks expensive. To perform operations in a PoW system, users must provide a proof of computation, whose production requires resource expenditure. This makes attacks like email spam and denial of service [37] prohibitively expensive, as they require many operations. However, honest users are also subject to these costs, and the system cannot balance deterring adversarial behavior but not honest one [43].

To circumvent this trade-off, Nakamoto [50] suggested introducing *internal rewards* for honest behavior (Table 1 summarizes this taxonomy). Indeed, nowadays PoW is widely used to secure decentralized and permissionless cryptocurrencies like Bitcoin and Ethereum [10]. These are replicated state-machines [12] that facilitate monetary ecosystems of internally-minted *tokens*, maintained by principals called *miners*. Miners that follow the protocol are rewarded with tokens; tokens are scarce, hence a market forms [45, 9]; so, miners can sell their obtained tokens, compensating them for their PoW expenses.

To guarantee their security [26, 15], PoW cryptocurrencies moderate their operation rate by dynamically tuning the required computation difficulty to match miner capabilities [40, 55]. Consequently, the PoW expenditure directly depends on token values [48, 4, 42] – higher token prices imply higher mining rewards, which draw more miners to participate, leading to more expended resources. This results with the external PoW expenditure matching the internal mining rewards, balancing honest participation overhead with high attack costs.

Indeed, with exponentially-growing token values, the amount of resources spent on PoW mining has also been growing accordingly[1]. Bitcoin PoW computations alone are responsible for about 0.3% of the global electricity consumption [14, 17], surpassing medium-sized countries like Colombia and Switzerland [23]. This level of resource guzzling is unsustainable [21, 60], bears a significant ecological impact [33, 49], and prevents adoption [54, 32]. Unfortunately, Nakamoto's mechanism directly incentivizes external expenditure at the same rate as of the internal rewards, and offers no means of reducing its external effects.

Previous work (§2) explored PoW alternatives for cryptocurrencies, notably focusing on *Proof of Stake* (*PoS*) [39, 29, 24]. Such systems avoid the external resource expenditure by replacing the computational effort with internal token ownership requirements. However, PoS systems operate, and are secured under, qualitatively-different assumptions. Namely, they rely on deprecated data deletion [39], or extended user availability [29, 30, 24]. Moreover, new users need to obtain tokens to participate, requiring the permission of current stakeholders.

We note that naive adjustments of the cryptocurrency minting rate do not reduce the external expenses; that a simple reduction of rewards hampers security; and that *forcing* miners to internally-spend breaks the permissionless property of the system. We review these options in our extended report [63].

[1] https://www.blockchain.com/charts/

In this work we present *Hybrid Expenditure Blockchain* (*HEB*), the first PoW protocol with lower external costs than its internal rewards. Despite the reduced external expenditure, *HEB* provides similar security guarantees against rational attackers compared to the more-wasteful Nakamoto protocol. *HEB* is tunable, allowing to optimize for desired properties.

***HEB* Overview.** The main challenge is to reduce the external expenses while keeping attack (and also participation) costs high. These objectives seem to contradict, as in previous work [19, 36, 50, 10] the participation costs are only external.

This is the main novelty of *HEB* – it is a generalization of Nakamoto's protocol that *enables* and *incentivizes* miners to forfeit system tokens as part of the mining process. Miners that do so increase their rewards, resulting with this being the optimal behavior. So, the external mining expenses in *HEB* are lower than existing PoW blockchains, while the total expenses (internal and external) are the same.

Similarly to Bitcoin, *HEB* constructs a tree data structure of elements named *blocks*, where the longest path defines the system state. Miners produce PoW to create blocks and broadcast them with a p2p network. However, unlike Bitcoin, *HEB* considers epochs of blocks, and the mining rewards for each epoch are distributed after its conclusion.

In *HEB* there are two types of blocks miners can create – *regular* or *factored*, which have a *weight* attribute with values 1 and $F > 1$, respectively. The epoch rewards are distributed among the miners proportionally to the relative weight of their blocks (in the epoch). Miners can always create regular blocks, but have to forfeit system tokens in epoch k to create factored blocks in epoch $k + 1$. This mechanism incentivizes miners to divert some of their total participation budget internally to create factored blocks, reducing their external PoW expenditure. The ratio between internal and external expenses is tuned with a parameter ρ, determining the expenditure required to create a factored block.

To maintain the total circulating token supply, the internally-expended tokens are distributed proportionally among all system entities (i.e., any token holder) at the epoch conclusion. This redistribution maintains the token value as in a standard PoW cryptocurrency (e.g., Bitcoin), as well as the relative wealth of all system entities. The internally-spent tokens are redistributed using a novel redistribution technique, which might be of independent interest (e.g., for regulating transaction fees, cf. [58]).

We emphasize that *HEB* draws ideas from PoS, prominently the utilization of system tokens for security, but the model assumptions, the solution, and the guarantees are distinct. In particular, *HEB* uses the standard PoW assumptions and miners expend (lose) their tokens for mining, whereas in PoS participants derive their power from maintaining ownership.

***HEB* Analysis.** Our goal in analyzing *HEB* is twofold: show it is incentive-compatible, secure, and permissionless, similarly to Nakamoto's protocol, and; show it achieves the desired, reduced external impact. For these two purposes we lay forth the following groundwork. We begin by modeling the cryptocurrency ecosystem, and follow with the underlying data structures, participants, and execution (§3). As in previous work [20, 28, 61], we consider a set of rational miners that optimize their revenues and an adversary who is willing to expend resources in order to attack.

We then instantiate *Nakamoto*'s protocol (§4) and use it for a comparison baseline, following with the formal presentation of *HEB* (§5).

To compare and contrast *HEB* with *Nakamoto* we consider a variety of cryptocurrency metrics (§6). These include common security metrics, namely coalition resistance and tendency to encourage coalitions [20, 59, 62]. We also introduce a new metric – *external expenses*, measuring the resources spent on PoW. Instead of the binary metric permissioned/permissionless (classical-consensus-protocols/Nakamoto-blockchain, respectively), we introduce the continuous metric of *permissiveness*, describing the cost of joining the system.

Finally, we tease apart the common safety-violation security metric into two. We observe that safety-violating chain-reorganization attacks [57, 5, 35] in existing PoW blockchains require high resource investment from the attacker; however, once successful, they completely refund themselves. We therefore consider this type of attacks, as well as a sabotage variant where the attacker is not refunded. We show that in *HEB* the attack cost for the refunded variant is linear in the total expenses (as secure as *Nakamoto*), and that the sabotage variant costs are linear in the external ones.

HEB includes several parameters for the system designer to tune. As an example, we present a specific choice of parameter values (§7). Choosing the prominent Bitcoin ecosystem as a reference point, we analyze *HEB* and show this parameter choice is practical, achieves strong security guarantees, and reduces the external consumption by half – the equivalent of reducing the entire electricity consumption of Denmark [14].

In summary, we expand the PoW design space by introducing internal expenses. We present *HEB* – a PoW blockchain protocol with external expenses that are lower than its internal rewards. We prove that *HEB* offers similar security guarantees against rational attackers compared to pure PoW solutions, and show it can significantly reduce the latter's ecological impact.

2 Related Work

In Nakamoto's blockchain and all subsequent PoW protocols we are aware of, the incentives equal the value of the generated cryptocurrency tokens (and fees). We are not aware of previous work tuning PoW expenditure in cryptocurrencies – the main focus of this work.

We proceed to survey PoS and analysis approaches. Our extended report [63] discusses permissioned and trusted hardware solutions that make qualitatively-stronger assumptions; protocols that expend different external resources rather than electricity, for which *HEB* applies equally well; and protocols with several types of internal tokens that do not achieve incentive compatibility nor reduced external expenses.

Proof of Stake. *HEB* and PoS are fundamentally different: the latter limits miner participation to those with stake in the system, i.e., miners who own tokens; the former does not. Moreover, in PoS the Sybil-deterrence [18] is due to the cost of acquiring and holding the system tokens, which the participants maintain throughout the system execution. In contrast, *HEB* relies on PoW, and the participants *spend* the internal currency.

PoS systems like Algorand [39], Ouroboros [29], Tezos [30] and the forthcoming Ethereum 2.0 [24] are designed and analyzed under different assumptions than PoW. Their security is based on users' owned tokens rather than their expended resources. They assume a new participant wanting to join the system can acquire (or, alternatively, lock as a collateral [39, 30, 24]) as many system tokens as she can afford. That is, existing system miners authorize transactions that introduce new system miners, even if these result in a state less favorable from their perspective. Additionally, to combat *long-range attacks* [16, 27] and *nothing-at-stake* [56, 8], these systems assume users voluntarily delete deprecated data [39], or assume users remain online for extended periods [29, 30, 24].

In contrast, *HEB* is PoW-based, and newly-joining miners do not require the cooperation of existing miners to join. It is also resistant to said long-range attacks and the nothing-at-stake problems, and hence does not rely on voluntary data deletion or user persistence.

A parallel work [25], which draws ideas from a previous version of this report, suggests emulating PoW over PoS. The main idea is that the stake used for the consensus degrades over time and usage, mimicking the external expenditures of PoW systems. However, as built atop of PoS, it also requires the aforementioned assumptions.

Proof of Work Analysis. We use the standard techniques [20, 59, 52, 44, 28] to analyze *HEB*'s security and incentive compatibility. The evaluation metrics used are a formalization of previous ones presented by said work, and also include definition of new ones regarding the external expenditure and permissiveness level. To the best of our knowledge we are the first to define, evaluate and optimize for such metrics.

Chen et al. [13] define and analyze desired properties of reward allocation schemes in PoW cryptocurrencies. Their work focuses on the reward of a single block, and does not consider environmental impact nor malicious miners. We note that *HEB*'s reward allocation rule is incentive-compatible and Sybil-resistant, satisfying those desired properties.

3 Model

We present a model for an abstract blockchain system, instantiated with a cryptocurrency protocol. This allows us to compare different instantiations, namely *Nakamoto* and *HEB*. We first define the monetary value of system tokens using an exogenous reference-point fiat currency (§3.1). We follow by presenting the blockchain, the participating entities and how the system derives its state (§3.2). We then define how a cryptocurrency protocol instantiates that system, defining an internal system currency based on the blockchain (§3.3), and explain how the system makes progress (§3.4).

3.1 Cryptocurrency Economics

The external expenditure of a PoW cryptocurrency system depends on the rewards it grants miners and on mining costs. We note that mining rewards are internal while PoW costs are external, hence we first define the relation, or the exchange rate, of the two.

The reward is an amount of the system's internal currency *ic* (e.g., Bitcoin, Ether), and the external cost is an amount of an external currency *ec* (e.g., USD). We assume the external currency has a market capital orders of magnitude larger than that of the internal currency[2], and it effectively represents real values.

We assume there is an instantaneous and commission-free exchange service of *ec* and *ic*, where the exchange rate matches token real value. To simplify presentation we normalize the price level so the exchange rate is one, and assume the exchange is available to all participating entities. We often sum *ec* and *ic*, meaning the sum of their values in real terms.

3.2 Blockchain and System Principals

The system comprises a shared *global storage*, a *scheduler*, and two types of *entities*: system *users*, and principals maintaining the system named *miners*.

The global storage is an append-only set containing elements called *blocks*. Each block includes a reference to another block and data generated by system entities, with the only exception being a so-called *genesis block* that contains neither. The global storage initially

[2] `https://fiatmarketcap.com/`

contains only the genesis block, thus defining a directed tree data structure. We refer to paths in the data structure starting at the genesis block as *chains*. We partition a chain C to *epochs* of ℓ blocks.

As common [20, 59, 52, 61] we assume that the sets of entities are static during an epoch execution, that is, entities do not join or leave during the course of an epoch.

Each miner i has a local storage accessible only to her, which, like the global storage, is an append-only block set. The scheduler invokes miners, allowing them to create blocks in their local storage, and to publish their local blocks by copying them to the global storage. We denote by N_i^k the number of blocks in epoch k created by miner i. For presentation simplicity, we assume the main chain at an epoch beginning remains a prefix of the main chain throughout the entire epoch. Note this does not rule out the main chain changing during an epoch, but only that its initial prefix does not.

Entities derive the *system state* by parsing the global storage according to the block order of the main chain. They might choose to infer the state based on a chain prefix, excluding potentially-volatile suffixes [26], such as in the case of multiple longest chains. Such considerations are outside the scope of this work.

3.3 Instantiating a Cryptocurrency Protocol

The system is instantiated with a cryptocurrency protocol Π that defines a currency internal to the system, *ic*. The protocol Π maps all its internal tokens to system entities with a function $Bal^{\Pi}(C)$, taking as input a chain C. The function returns a vector where each element $Bal_i^{\Pi}(C)$ is the number of tokens mapped to entity i. When the context is clear, we often omit the protocol name Π and simply write $Bal(C)$.

We say the total value of tokens mapped to an entity is her *ic balance*, and note the total number of tokens is the sum of all balances. The protocol mints $r_k \cdot \ell$ new tokens at the end of each epoch k, and we often omit the epoch index when it is clear from context. This means the number of tokens is fixed throughout any epoch k, and increases when epoch $k+1$ begins. The protocol Π maps the newly-minted tokens to entities using $Bal(C)$.

Aside from their owned *ic*, miners also own *ec*. We use the terms *internal* and *external balances* to distinguish the different holdings, and simply *balance* for their aggregate value.

Miners expend all their balance on system maintenance. In practice, a principal can split its balance, using some of it as a miner, and the rest as a user. We model such principals as two separate entities – a miner that spends all its balance, and a user that holds the rest.

For any epoch k we denote by $B_i^{ec}(k)$, $B_i^{ic}(k)$ and $B_i(k)$ the initial external, internal and total balances of each miner i, respectively. We also denote by $b_i^{ec}(k)$, $b_i^{ic}(k)$ and $b_i(k)$ the *relative* (out of all miners' balances) the external, internal and total balances of miner i. Finally, we denote the total balances of all miners by $B_{Miners}(k)$ and of all users by $B_{Users}^{ic}(k)$.

We assume the value of expended resources by the miners on system maintenance in a single epoch k is much smaller than the system market cap. That is, the balance of all miners is negligible compared to that of all users, i.e., $B_{Miners}(k) \ll B_{Users}^{ic}(k)$. In practice, $\frac{B_{Miners}(k)}{B_{Users}^{ic}(k)} \ll 10^{-7}$ holds for both Bitcoin and Ethereum[3].

[3] `https://coinmarketcap.com/currencies/`

3.4 Execution

Initially, the global storage contains only the genesis block, and each miner has an empty local storage. The state variables (like the global and local storage) change over time, but we omit indexing as it is clear from the context.

The system progresses is orchestrated by a scheduler, where epoch k begins when the main chain is of length ℓk. First, the scheduler lets miners set their internal and external balances using the exchange service, achieving their preferred balance of the two. We use the term *allocate* to describe this action, and say miner i allocates her balance $B_i(k)$ with the invocation of the $\mathsf{Allocate}_i(B_i(k))$ function, returning a tuple of her internal and external balances $\langle B_i^{ic}(k), B_i^{ec}(k)\rangle$.

Note that modeling changes in the miner set and balance allocations at epoch transitions is for presentation simplicity; these occur throughout the system execution.

The rest of the epoch execution progresses in steps, until the main chain is extended by ℓ blocks. Each step begins with the scheduler selecting a single miner at random, proportionally to her relative *external expenditure*, that is $\Pr(\text{scheduler selects } i) = b_i^{ec}(k)$. Similarly to previous work [20, 1, 59, 52], these steps represent a standard PoW mechanism and its logical state changes, and entities have synchronous access to the global storage.

The scheduler invokes the selected miner i's function $\mathsf{Generate}_i^{\Pi}()$, returning a newly generated block, and adds it to miner i's local storage. The protocol Π states block validity rules in $Bal^{\Pi}(C)$, and invalid blocks do not affect the system state. Creating an invalid block or not creating one at all is sub-optimal and we only consider miners who avoid doing so.

Next, the scheduler lets any miner i publish any of her unpublished blocks by invoking $\mathsf{Publish}_i()$, returning a subset of her previously-private local blocks. The scheduler adds the returned blocks to the global storage, and repeats this process until all miners do not wish to publish any more blocks. This captures strategic-block-release behaviors [20, 59, 52].

The cryptocurrency protocol Π includes implementations of $\mathsf{Allocate}_i(B_i(k))$, $\mathsf{Generate}_i^{\Pi}()$, and $\mathsf{Publish}_i()$ that each miner i should follow. We refer to the tuple of three implementations as the *prescribed strategy* and denote it by $\sigma^{\Pi}_{prescribed}$. The protocol Π is therefore a tuple of the balance function Bal^{Π} and a prescribed strategy $\sigma^{\Pi}_{prescribed}$. Note that Π cannot force miners to follow $\sigma^{\Pi}_{prescribed}$.

4 Nakamoto's Protocol

As an example and to serve as a baseline, we instantiate an epoch-based *Nakamoto* protocol (used with $\ell = 1$ in Bitcoin, Litecoin, etc.) in our model.

The balance function of *Nakamoto* awards each miner i with r tokens per block she created in the epoch, and a total of $\ell \cdot r$ new tokens are minted. Hence, the balance of each miner i at epoch conclusion is $N_i \cdot r$.

The prescribed strategy $\sigma^{Nakamoto}_{prescribed}$ states that each miner i allocates her balance $B_i^{ec} = B_i$ and $B_i^{ic} = 0$, extends the longest chain, and publishes her blocks immediately. In case of multiple longest chains, $\sigma^{Nakamoto}_{prescribed}$ picks uniformly-at-random[4].

[4] Bitcoin defines a different tie-breaking rule – pick the first longest chain the miner became aware of. Therefore its security guarantees vary, depending on the underlying network assumptions. As in previous work [41, 59], we avoid such assumptions by considering the uniformly-at-random variation.

5 *HEB* Protocol

We are now ready to present *HEB*. Briefly, it incentivizes miners to expend their balances internally by enabling miners who do so to create higher-reward blocks. Two parameters, $\rho \in [0,1)$ and $F \in \mathbb{R}_{>1}$, control the reward distribution mechanism. We detail the different block types, the reward distribution mechanism, and the desired strategy.

Block types. Each block has a type, determined at its creation – either *regular* or *factored*. During an epoch, miner i can create regular blocks whenever the scheduler invokes $\mathsf{Generate}_i()$. However, aside from an invocation by the scheduler, creating a factored block requires an internal expenditure of $\rho \cdot r$ in ic by miner i at the previous epoch; recall we model these internal expenditures as if they occur at the start of current epoch. We emphasize that creating blocks of either type occurs only by an invocation of the scheduler, i.e., based on the external expenditure of the miner.

Consequently, if $\rho > 0$ then miner i can create at most $\left\lfloor \frac{B_i^{ic}}{\rho \cdot r} \right\rfloor$ factored blocks in an epoch on chain C. *HEB* assigns a *weight* to each block according to its type, and factored and regular blocks have weights of F and 1, respectively.

Reward distribution. *HEB* distributes the ℓr minted tokens among the miners in proportional to their block weights. Denote by $W_i(C)$ the total block weight of miner i on chain C. So, miner i gets $\frac{W_i(C)}{\sum_j W_j(C)} \ell r$ tokens for her created blocks.

The internal expenses B_{Miners}^{ic} are distributed among all system entities (i.e., including users) proportionally to their ic balances at the epoch beginning. So, miner i receives $\frac{B_i^{ic}}{B_{Miners}^{ic} + B_{Users}^{ic}} B_{Miners}^{ic}$ tokens from the redistribution. We discuss shortcomings of other distribution schemes in the extended report [63].

In summary, miner i gets $\frac{W_i()}{\sum_j W_j()} \ell r + \frac{B_i^{ic}}{B_{Miners}^{ic} + B_{Users}^{ic}} B_{Miners}^{ic}$ at the epoch conclusion.

Prescribed strategy. The prescribed strategy $\sigma_{prescribed}^{HEB}$ states that miners allocate their balance with ratio ρ and create factored blocks up to their internal balance limitation. Formally, miner i allocates $B_i^{ic} = \rho B_i$ and $B_i^{ec} = (1-\rho) B_i$. If $\rho = 0$ then the miner creates all blocks as factored, and if $\rho > 0$ then only the first $\left\lfloor \frac{B_i^{ic}}{\rho \cdot r} \right\rfloor$ are factored. As in *Nakamoto*, miner i points her created blocks to the longest chain, and publishes them immediately.

▶ **Note.** *Setting $\rho = 0$ enables miners to create all blocks as factored, and setting $F = 1$ removes motivation to create any factored blocks at all. In both cases there is only one practical block type, reducing HEB to Nakamoto.*

We discuss practical implementation aspects of *HEB* in our extended report [63] – shortening epochs, utilizing a pure PoW ramp-up period to create a sufficiently-large currency circulation, and addressing discretization issues.

6 Evaluation

We now evaluate *HEB*, showing how parameter choices affect its properties. For that, we formalize the cryptocurrency system as a game played by the miners, striving to maximize their rewards (§6.1). We use *Nakamoto* as a baseline, highlighting *HEB* parameter choices that result with significantly lower PoW expenditure while limiting undesirable side-effects.

To compare we first need to define criteria. Hence, throughout this section we present cryptocurrency evaluation metrics, each followed by its evaluation in *Nakamoto* and in *HEB*. We consider common security metrics [20, 13, 26] regarding the *incentive compatibility* of a system (§6.2 and §6.3); refine the common safety-violation security metric [38, 7, 46], measuring attack *costs* (§6.4); generalize the binary permissioned/permissionless notion [50, 3] to a continuous metric (§6.5); and conclude with a new metric for external expenses (§6.6).

6.1 Block Creation as a Game

The model gives rise to a game, played by miners for the duration of a single epoch k (hereinafter omitting the epoch index). We define the utility U_i of miner i as her expected cryptocurrency holdings at the conclusion of the epoch.

As commonly done in the analysis of cryptocurrency protocols [20, 34, 61, 59], we assume that during an epoch the system is quasi-static, where all miners participate and the total profit is constant. In operational systems miners participate for a positive profit [48, 64], but discussing the required return-on-investment ratio for such behavior is out the scope of this work, and we arbitrarily assume it to be 0 [22, 31]. Accordingly, the sum of all miner utilities equals the overall miner balances, that is, $B_{Miners} = \sum_i U_i$.

We normalize the number of newly-minted tokens per block to be one, meaning $r = 1$, so a total of ℓ tokens are created in the epoch.

The strategy space comprises choosing the allocation ratio, what blocks to generate, and when to publish them, i.e., implementations of Allocate (), Generate () and Publish ().

▶ **Example** (Nakamoto)**.** We demonstrate the compatibility of our definitions and modeling with previous results [51] regarding *Nakamoto*. We consider a scenario where all miners follow $\sigma^{Nakamoto}_{prescribed}$. So, all miner balances are in *ec* and consequently $b_i = b_i^{ec}$. Additionally, all miners extend the longest chain.

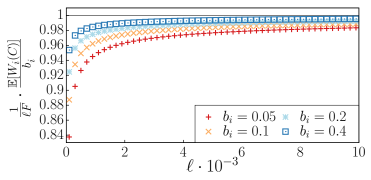
We note the scheduler picks at each step a miner proportional to her relative external balance. We can consider each pick as a Bernoulli trial where miner i is picked with success probability of b_i. So, the number of blocks a miner i creates in an epoch is binomially distributed $N_i \sim \text{Bin}\left(\ell, b_i\right)$.

Therefore, $\mathbb{E}\left[N_i\right] = b_i^{ec}\ell$, and the utility of miner i is $U_i^{Nakamoto} = b_i^{ec}\ell$, matching previous analysis [50]. Summing for all miners yields $B_{Miners} = B^{ec}_{Miners} = \ell$, and the expected cost to create each block is 1, matching its reward.

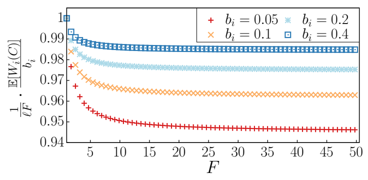
6.2 *Size bias*

Cryptocurrency security relies on having multiple, independent miners, none of which has control over the system [13, 26]. For that, these systems strive to distribute their rewards in a way that is *size-indifferent*, meaning that miners get relative reward matching their relative balances, and hence have no incentive to coalesce. The metric *Size bias* measures how well a protocol satisfies this desideratum when all miners follow the prescribed strategy. Unlike the other metrics, it is evaluated for a specific balance distribution.

Formally, assume a balance distribution and that each miner i with relative balance b_i follows $\sigma^{\Pi}_{prescribed}$. The utility of such miner is U_i^{Π}, and her relative utility is $\frac{U_i^{\Pi}}{\sum_j U_j^{\Pi}}$. We define *Size bias* to be the maximal difference of each miner's relative balance and relative utility, that is, *Size bias* $\triangleq \max_i \left| b_i - \frac{U_i^{\Pi}}{\sum_j U_j^{\Pi}} \right|$.



(a) $F = 20$.



(b) $\ell = 1000$.

Figure 1 $\frac{1}{\ell F} \cdot \frac{\mathbb{E}[W_i(C)]}{b_i}$ for ρ and F values.

Systems strive for *Size bias* to be minimal, as higher values indicate more disproportionate shares. Preferably, *Size bias* $= 0$, indicating all miners are rewarded proportionally.

In practice, there is an inherent advantage for having a larger relative balance due to fork-rate [53, 47] and economy-of-scale [1] considerations, and although *Size bias* $= 0$ is a theoretical desideratum, systems like Bitcoin successfully operate even with non-zero values.

Nakamoto. Recall that in *Nakamoto* the utility of each miner i is $U_i^{Nakamoto} = b_i^{ec}\ell$, meaning $\frac{U_i^{Nakamoto}}{\sum_j U_j^{Nakamoto}} = b_i$ and in our model *Size bias* $= 0$. This matches previous analysis [50].

HEB. We move to analyze *HEB*'s *Size bias*. Throughout this section we present a summary of the analysis and its results, and bring the details in our extended report [63].

In *HEB* it holds that if all miners follow the prescribed strategy then $B_{Miners} = \ell$ and the utility of each miner i is $U_i^{HEB} = \frac{\mathbb{E}[W_i(C)]}{\sum_j \mathbb{E}[W_j(C)]}\ell$. To show the above we first show the redistributed internal currency a miner receives is negligible, allowing us to focus on the minting reward. For that we analyze the number of blocks a miner creates. Then, we derive her conditional total block weight, that is, her total block weight conditioned on the number of blocks she creates. We proceed to derive her expected total block weight, and conclude with finding her utility.

Then, we show that *HEB* achieves *Size bias* $= 0$ with sufficiently long epochs, as formalized by the following corollary:

▶ **Corollary 1.** *HEB achieves* $\lim_{\ell \to \infty}$ *Size bias* $= 0$.

The proof begins by showing that if all miners follow the prescribed strategy, then for any two miners i, j the ratio of the expected weight and relative budget is equal $\frac{\mathbb{E}[W_i(C)]}{b_i} = \frac{\mathbb{E}[W_j(C)]}{b_j}$ iff *Size bias* $= 0$. Then it shows that if all miners follow $\sigma^{HEB}_{prescribed}$, then $\lim_{\ell \to \infty} \frac{\mathbb{E}[W_i(C)]}{b_i} = \ell F$ for any miner i, and consequently, the former condition holds.

We conclude with concrete number instantiations, showing that *Size bias* improves (decreases) with longer epochs (larger ℓ) and a smaller factor (smaller F) value, while being independent of ρ. We also show more balanced distributions have lower *Size bias*, but note these are not under the control of the system designer. Considering practical parameter choices, we show that even for an extreme balance distribution, *HEB* achieves *Size bias* $< 0.3\%$. In a similar, yet balanced scenario, *Size bias* $= 0$.

For that, we calculate $\frac{\mathbb{E}[W_i(C)]}{b_i}$ for various F, ℓ and b_i values. We present the results in Fig. 1, multiplied by $\frac{1}{\ell F}$ for comparison purposes. Although we present results for specific configurations, we assert that different parameter values yield the same qualitative results.

Table 2 *Size bias* for $\ell = 1000$ and $F = 20$.

Balance distribution					*Size bias*
b_1	b_2	b_3	b_4	b_5	
0.20	0.80	–	–	–	**0.0029**
0.10	0.15	0.20	0.20	0.35	**0.0025**
0.20	0.40	0.40	–	–	**0.0015**
0.20	0.20	0.30	0.30	–	**0.0007**
0.20	0.20	0.20	0.20	0.20	**0.0000**

Fig. 1a shows for a fixed $F = 20$ the value of $\frac{1}{\ell F} \cdot \frac{\mathbb{E}[W_i(C)]}{b_i}$ as a function of ℓ. As expected, $\frac{1}{\ell F} \cdot \frac{\mathbb{E}[W_i(C)]}{b_i}$ approaches 1 as ℓ grows, leading towards *Size bias* $= 0$. However, for any fixed ℓ value, miners of different b_i have different $\frac{1}{\ell F} \cdot \frac{\mathbb{E}[W_i(C)]}{b_i}$, resulting with *Size bias* > 0.

We also illustrate the effect of F on $\frac{1}{\ell F} \cdot \frac{\mathbb{E}[W_i(C)]}{b_i}$. Fig. 1b shows $\frac{1}{\ell F} \cdot \frac{\mathbb{E}[W_i(C)]}{b_i}$ for $\ell = 1000$ as function of F. At the region of lower F values, increasing F also increases the difference of $\frac{1}{\ell F} \cdot \frac{\mathbb{E}[W_i(C)]}{b_i}$ for different b_i. However, as F becomes larger, then $\frac{1}{\ell F} \cdot \frac{\mathbb{E}[W_i(C)]}{b_i}$ tends towards a constant and the difference for different b_i remains fixed. This is expected, as for larger F values the expected weight is dominated by the expected weight of factored blocks (see [63]), and the expected weight becomes linear in F.

We dedicate the rest of this section to analyze how different balance distributions affect miners' utilities and *Size bias*. We consider various settings of at most 5 miners with epoch length of $\ell = 1000$ blocks and $F = 20$.

For each setting we numerically calculate *Size bias* and present it, along with its respective balance distribution, in Table 2. We choose these specific settings to demonstrate *Size bias* both balanced and extreme distributions.

Table 2 shows that more extreme balance distributions results in higher *Size bias*. For instance, consider the setting with only two miners where $b_1 = 0.2$ and $b_2 = 0.8$. This setting leads to the highest value of *Size bias* $= 0.0029$. Note that this is an unrealistic setting, presented only as an example for a highly-uneven distribution. Even in this extreme scenario miner 1 has a degradation of less than 0.3% in her relative utility. More balanced settings lead to lower *Size bias* values.

In summary, even a highly-unbalanced distribution results in minor deviations from proportional rewards. Increasing ℓ and decreasing F both reduce these deviations.

6.3 *Nash threshold*

Recall protocol Π provides a prescribed strategy $\sigma^{\Pi}_{prescribed}$ that miners individually choose whether to follow. The protocol properties rely on miners following this strategy [20, 59, 52, 11, 62, 28, 61], hence it should incentivize miners to do so.

The question is whether the prescribed strategy is a Nash-equilibrium, meaning no miner can benefit from individually deviating to a different strategy. Like in previous work [20, 59, 52], the *Nash threshold* metric is the maximal relative miner balance that achieves this: If all miners have relative balances smaller than the threshold, then the prescribed strategy is a Nash-equilibrium.

Formally, denote by $\sigma^{\Pi}_{\mathrm{i},best}$ the best-response strategy of miner i with relative balance b_i when all other miners follow $\sigma^{\Pi}_{prescribed}$. *Nash threshold* is the maximal value b_i such that $\sigma^{\Pi}_{\mathrm{i},best} = \sigma^{\Pi}_{prescribed}$. It follows that $\sigma^{\Pi}_{prescribed}$ is a Nash-equilibrium if all miner relative balances are not greater than *Nash threshold*.

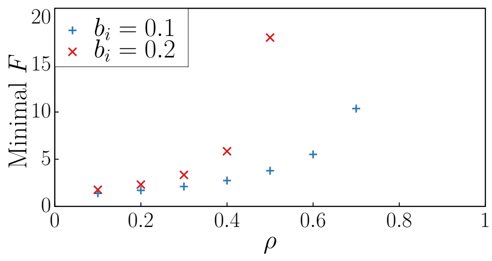


Figure 2 Minimal F for *Nash threshold*.

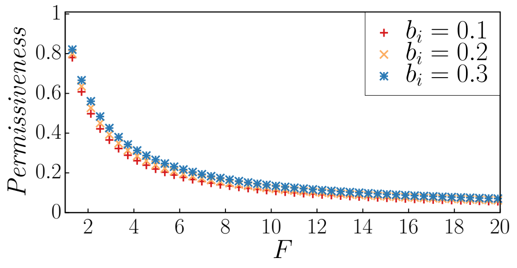


Figure 3 *HEB Permissiveness*.

Nakamoto. Sapirshtein et al. [59] showed that for *Nakamoto* with the uniform tie-breaking fork selection rule (see §4) the metric value is *Nash threshold* $= 0.232$.

HEB. An optimal miner strategy must consider how to allocate the balance, which previous blocks to point to, what block type to create, and when to publish created blocks.

Before considering the best strategy a miner can follow, we start by considering a specific, natural, *PoW-only* strategy $\sigma^{HEB}_{PoW\text{-}only}$, which simply ignores the internal expenditure. The idea of $\sigma^{HEB}_{PoW\text{-}only}$ is to maximize the block creation rate by expending all resources externally. The miner tries to create all the epoch blocks herself, and thus obtain all the epoch rewards.

This strategy is of interest as it abuses the internal expenditure mechanism; it is also simple enough to lend itself to a closed-form analysis. Specifically, we show (in our extended report [63]) that this strategy is more rewarding than $\sigma^{HEB}_{prescribed}$ if $b_i > \frac{1-\rho}{2-\rho}$. It follows the latter is an upper bound for *Nash threshold*. We note that higher ρ values lower the bound, as miner i is competing against less external balance. This result matches *Nakamoto*, as if $\rho = 0$ then $\frac{1-\rho}{2-\rho} = 0.5$, yielding the established 50% bound [59, 42]. We now move to search for the best-response strategy of a miner.

Following previous work [59, 28], we use *Markov Decision Process* (*MDP*) to search for the optimal strategy in *HEB*. The MDP includes the internal expenditure and block weights, and produces miner i's best-response strategy $\sigma^{HEB}_{\mathrm{i},best}$ based on system parameters. We defer the MDP technical details to our extended report [63].

We note the state and action spaces grow exponentially with the epoch length, limiting available analysis to relatively small epoch values. Therefore, similarly to previous work [59, 28], we also limit the state space by excluding strategies requiring longer, and thus less probable, sequences of events.

Our focus is finding the required parameter values for which $\sigma^{HEB}_{\mathrm{i},best}$ matches $\sigma^{HEB}_{\mathrm{i},prescribed}$. Recall that $\sigma^{HEB}_{\mathrm{i},best}$ is the optimal implementation of $\mathsf{Allocate}_i()$, $\mathsf{Generate}_i()$ and $\mathsf{Publish}_i()$ given B_i and the system parameters ℓ, F, ρ, hence we take the following approach.

We fix $\ell = 10$ to limit the state space, and for various values of ρ and b_i we use binary-search to find the minimal $F \in [1, 10^8]$ value such that $\sigma^{HEB}_{\mathrm{i},best} = \sigma^{HEB}_{prescribed}$. First, we consider $\mathsf{Allocate}_i()$ implementations that let miner i create a natural number of blocks (allocating balance to enable the creation of a fraction of a block is strictly dominated, enabling discretization of possible implementations). For each such implementation we use the MDP to obtain the optimal implementation of $\mathsf{Generate}_i()$ and $\mathsf{Publish}_i()$. We let the miner play the resultant strategies, and take the most rewarding to be $\sigma^{HEB}_{\mathrm{i},best}$.

We present the results in Fig. 2, showing that increasing F values and lowering ρ increases *Nash threshold*. Specifically, for $b_i = 0.2$ the required F values grow exponentially with ρ up to $\rho = 0.5$, and from there even the maximal F value does not accommodate the desired behavior. We note a similar behavior for $b_i = 0.1$, growing exponentially with ρ up to $\rho = 0.7$, being the maximal ρ that leads to $\sigma^{HEB}_{prescribed}$ being a Nash-equilibrium.

We also note that lower b_i requires lower F values, and specifically, there are no F and ρ values for which the configuration of $b_i = 0.3$ achieves a Nash-equilibrium. This is expected as the profitability threshold for selfish-mining variants is $b_i = 0.232$ [59], and indeed the resultant best-response strategies resemble selfish-mining in *Nakamoto*.

We conclude that *Nash threshold* relies on ℓ, F and ρ; by setting $F = 20$, we can obtain *Nash threshold* $= 0.2$ even for $\rho = 0.5$, close to *Nakamoto*'s value [59].

6.4 *Free safety-violation threshold* and *Safety-violation threshold*

We consider safety-violation attacks [38, 7, 46, 2, 57, 5, 35] as scenarios where an attacker causes the system to make an invalid transition. This can be achieved by creating and publishing an alternative chain, surpassing the main one. The original chain blocks are then discarded, and the system state is reinstated according to the blocks on the alternative chain.

To mount this attack in *Nakamoto* the attacker expends her resources on creating blocks to form the alternative chain; recall that each block costs its worth in reward to create. Therefore, if the attack is successful, the attacker is fully compensated for her expenditures by the rewards from her created blocks. As such, there is a threshold of required resources to mount this attack, but once met, the attack is *free*.

The metric *Free safety-violation threshold* measures the minimal required balance for a miner to deploy such a refunded safety-violation attack on the system, assuming all other miners follow the prescribed strategy $\sigma^{\Pi}_{prescribed}$. As shown in previous work [7], the attacker may rent vast computational resources for a short period of time or a moderate amount for longer periods. We therefore measure the expected cost to create a single block, disregarding the attack duration and amplitude.

The *Safety-violation threshold* metric removes the refund requirement, and simply represents the cost to create a block.

Formally, assume all miners follow $\sigma^{\Pi}_{prescribed}$. Then, *Free safety-violation threshold* is the minimal cost to create a block, guaranteeing full compensation should it be on the main chain, and *Safety-violation threshold* is this cost without any further compensation guarantees.

Nakamoto. In *Nakamoto* the cost to create each block is 1, hence *Free safety-violation threshold* $= 1$. All blocks produce the same reward, hence a miner cannot reduce the cost for a safety-violation attack by choosing to create less-rewarding blocks. Therefore, *Safety-violation threshold* $= 1$.

HEB. In equilibria the total external expenses are $1 - \rho$ of the total balances, that is $B^{ec}_{Miners} = (1 - \rho)\, B_{Miners}$. As $B_{Miners} = \ell$ it follows that the required external expenses to create a single block is $1 - \rho$.

As other miners create factored blocks, a miner also has to create a factored block to be fully compensated for her expenses, requiring additional spending of ρ. Hence, the cost to create a single block is 1, so *Free safety-violation threshold* $= 1$. That is, *HEB* is as resilient to refunded attacks as *Nakamoto*.

Alternatively, a miner can disregard compensation and choose to create regular blocks, baring no additional internal expenses, and so *Safety-violation threshold* $= 1 - \rho$, which is less secure than *Nakamoto*. However, the lack of direct compensation makes these attacks very expensive, hence they are only available to a well-funded adversary with an exogenous utility, e.g., interested in destabilizing or short-selling a cryptocurrency.

Indeed, previous attack instances [57, 5, 35] were on relatively-small systems and were of the former, refunded type. We are not aware of such sabotage attacks happening in practice; this is possibly because the required expenditure surpasses the potential profit [2].

6.5 *Permissiveness*

Cryptocurrency protocols implement their own reward distribution mechanisms [13], and may choose to condition rewards on a miner having the internal system currency *ic*. For example, in PoS systems [39, 29], owning *ic* is a requisite, and miners without *ic* cannot participate and get rewards. In contrast, in PoW systems this is not the case.

Acquisition of *ic* involves an update of the new currency ownership in the system state. This requires the cooperation of the present system miners: They decide which state updates occur when placing data in their created blocks. So, if token ownership is a mining requirement, then a new miner wishing to participate requires the cooperation of existing miners.

Previous work considered either permissioned systems that require token ownership [29, 39, 30] (some also require explicitly locking owned tokens as a collateral), or permissionless systems [50, 10] that do not.

We generalize this binary differentiation to a continuous metric, *Permissiveness*, measuring the revenue of a newly-joining miner without cooperation from the incumbents. The metric is the ratio between the revenues of a miner where she failed or managed to obtain *ic*.

Formally, consider a miner i with balance B_i, and assume that all other miners follow $\sigma^{\Pi}_{prescribed}$. Denote by $\sigma^{\Pi}_{prescribed\text{-}no\text{-}ic}$ a strategy identical to $\sigma^{\Pi}_{prescribed}$ with the exception that the $\mathsf{Allocate}()$ implementation returns $\langle 0, B_i \rangle$. Note this captures the inability of miner i to obtain *ic*. Denote by $U^{\Pi}_{i,prescribed\text{-}no\text{-}ic}$ and by $U^{\Pi}_{i,prescribed}$ the utility of miner i if she follows $\sigma^{\Pi}_{prescribed\text{-}no\text{-}ic}$ and $\sigma^{\Pi}_{prescribed}$, respectively. We then define $\mathit{Permissiveness} \triangleq \frac{U^{\Pi}_{i,prescribed\text{-}no\text{-}ic}}{U^{\Pi}_{i,prescribed}}$.

If $\mathit{Permissiveness} = 1$ then a miner's utility is not affected by her inability to obtain *ic*, meaning the protocol is permissionless. In contrast, $\mathit{Permissiveness} = 0$ indicates that a miner who cannot obtain *ic* is completely prevented from participation.

Nakamoto. As a pure PoW blockchain protocol, *Nakamoto* miners do not require *ic* balance, so $\mathit{Permissiveness} = 1$.

HEB. Calculating both utilities, we get that $\mathit{Permissiveness} = (b_i + F(1 - b_i))^{-1}$ ([63]), which Fig. 3 presents for different values of b_i as a function of F. It shows that higher factor values F lead to lower *Permissiveness* values, making the system more permissioned. It also shows that miners with higher relative balances are slightly less susceptible to these effects.

Although failure to obtain *ic* results with a lower reward, it still enables the new miner to create blocks herself, removing the requirement for cooperation from the incumbents in the subsequent epochs. The reduced reward in the first epoch is a one-time cost that is negligible for a long-running miner. This is a significant and qualitative improvement over permissioned systems, where a miner that cannot obtain tokens [29] or lock them as a collateral [39, 30, 24] is blocked from all future participation.

6.6 *External expenses*

External expenses evaluates the external expenditure of the protocol, and lower values indicate a lower environmental impact. Formally, assume all miners follow $\sigma^{\Pi}_{prescribed}$. *External expenses* is the total of miner external expenses, measured in *ec*, normalized by the epoch length, i.e, $\mathit{External\ expenses} \triangleq \frac{B^{ec}_{Miners}}{\ell}$.

Nakamoto. The total miner expenses are $B^{ec}_{Miners} = \ell$, so $\mathit{External\ expenses} = 1$.

HEB. When all miners follow $\sigma^{HEB}_{prescribed}$ then $B^{ec}_{Miners} = (1-\rho)\, B_{Miners}$ and $\mathit{External\ expenses} = 1-\rho$. This is the main advantage of *HEB* over *Nakamoto*.

7 Practical Parameters

As we have seen, *HEB* presents several knobs for the system designer. Longer epoch length ℓ improves *Size bias*, however, also means that reward distribution takes longer. Higher factored block weight F improves *Nash threshold* at the expense of *Permissiveness*. Higher internal expenditure rate ρ reduces the external expenditures, but makes the system less robust against rational miners, and reduces the required costs for sabotage attacks.

The choice of parameter values should be according to the desired system properties. Each system has different goals, and we emphasize that determining optimal parameter values is not a goal of this work. Nevertheless, in this section we consider a specific parameter choice. We compare this instantiation to Bitcoin, and use the latter's miner balance distribution [6] as a representative example.

We choose the external cost parameter to be $\rho = 0.5$, the epoch length to be $\ell = 1000$, and the factor to be $F = 20$. First and foremost, this setting results with only half of the external resource consumption ($\mathit{External\ expenses} = 0.5$), which is equivalent to reducing the entire power consumption of Denmark [14]. This choice incentivizes rational miners with up to 0.2 relative balance to follow the prescribed strategy ($\mathit{Nash\ threshold} = 0.2$) down from Bitcoin's 0.232 [59]. Note that the largest miner in Bitcoin has a relative balance of 0.16 [6], so rational miners would follow the prescribed, honest mining behavior.

With Bitcoin's expected block creation interval of 10 minutes, having epochs of $\ell = 1000$ means mining rewards are distributed on a weekly basis. This is longer than the seventeen hours Bitcoin miners wait today, but arguably still an acceptable time frame.

The threshold for a refunded safety-violation ($\mathit{Free\ safety\text{-}violation\ threshold} = 1$), is as in Bitcoin, but the non-refunded variation is twice as cheap ($\mathit{Safety\text{-}violation\ threshold} = 0.5$). We note that we are not aware of attacks of either type on prominent cryptocurrency systems, and that the non-refunded type is unlikely due to the lack of endogenous compensation (§6.4).

In regards to permissiveness, a miner with 10% budget that fails to obtain any *ic* due to incumbent censorship is expected to get 5% of what she would have had with *ic* ($\mathit{Permissiveness} = 0.05$). Recall this is a one-time entry cost (§6.5), and a qualitative improvement on a permissioned system.

Finally, for the current Bitcoin miner balance distribution [6] the maximal relative advantage from size differences is 0.1% ($\mathit{Size\ bias} = 0.001$). We consider modifications to further decrease this value in our extended report [63].

8 Conclusion

We propose a new PoW paradigm that utilizes internal expenditure as a balancing mechanism for the external impact. We present *HEB* – a generalization of Nakamoto's protocol that allows its designer to tune external resource expenditure. We formalize evaluation metrics including a blockchain's resilience to sabotage and revenue-seeking attacks and permissiveness on a continuous scale. We propose practical parameters based on Bitcoin's ecosystem that cut down by half the PoW expenditure (equivalent to reducing the power consumption of an entire country) while maintaining similar security guarantees against practical attacks.

Natural questions that arise from the introduction of *HEB* are what should be the security target for cryptocurrency protocols, how to set the parameters dynamically, and how to govern them [30]. Beyond these, *HEB* extends the design space of decentralized systems, and is a step forward in realizing secure PoW systems with a sustainable environmental impact.

References

1 Nick Arnosti and S Matthew Weinberg. Bitcoin: A natural oligopoly. *arXiv*, 2018. `arXiv:1811.08572`.
2 Shehar Bano et al. Sok: Consensus in the age of blockchains. In *AFT*, 2019.
3 Mathieu Baudet et al. State machine replication in the libra blockchain, 2018.
4 Jörg Becker et al. Can we afford integrity by proof-of-work? In *WEIS*, 2012.
5 Bitcoin.com. Etc team finally acknowledges the 51% attack on network, 2020.
6 blockchain.info. Hashrate distribution, 2020. URL: `https://tinyurl.com/55nb9w2v`.
7 Joseph Bonneau. Why buy when you can rent? In *FC*, 2016.
8 Joseph Bonneau et al. Research perspectives on Bitcoin and second-generation cryptocurrencies. In *Symposium on Security and Privacy*, 2015.
9 Timothy C Brock. Implications of commodity theory for value change. In *Psychological foundations of attitudes*, pages 243–275. Elsevier, 1968.
10 Vitalik Buterin. Ethereum whitepaper, 2013.
11 Miles Carlsten et al. On the instability of bitcoin without the block reward. In *CCS*, 2016.
12 Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.
13 Xi Chen et al. An axiomatic approach to block rewards. In *AFT*, 2019.
14 Alex de Vries. Bitcoin's energy consumption is underestimated: A market dynamics approach. *Energy Research & Social Science*, 2020.
15 C. Decker and R. Wattenhofer. Information propagation in the Bitcoin network. In *P2P*, 2013.
16 Evangelos Deirmentzoglou, Georgios Papakyriakopoulos, and Constantinos Patsakis. A survey on long-range attacks for proof of stake protocols. *IEEE Access*, 7:28712–28725, 2019.
17 digiconomist. Bitcoin energy consumption, 2021. URL: `https://tinyurl.com/ehvat7fz`.
18 John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, 2002.
19 C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Crypto*, 1992.
20 Ittay Eyal and Emin Gün Sirer. Majority is not enough. In *FC*, 2014.
21 Peter Fairley. Blockchain world-feeding the blockchain beast if bitcoin ever does go mainstream, the electricity needed to sustain it will be enormous. *IEEE Spectrum*, 2017.
22 Amos Fiat et al. Energy equilibria in proof-of-work mining. In *EC*, 2019.
23 Cambridge Centre for Alternative Finance. Bitcoin electricity consumption index, 2019.
24 Ethereum Foundation. Ethereum 2, 2019. URL: `https://tinyurl.com/srr67va`.
25 Chaya Ganesh et al. Virtual asics: Generalized proof-of-stake mining in cryptocurrencies. ePrint, 2020.
26 Juan A. Garay et al. The Bitcoin backbone protocol. In *Eurocrypt*, 2015.
27 Peter Gaži et al. Stake-bleeding attacks on proof-of-stake blockchains. In *CVCBT*, 2018.
28 Arthur Gervais et al. Security and performance of proof of work blockchains. In *CCS*, 2016.
29 Yossi Gilad et al. Algorand. In *SOSP*, 2017.
30 LM Goodman. Tezos: a self-amending crypto-ledger white paper, 2014.
31 Guy Goren and Alexander Spiegelman. Mind the mining. In *EC*, 2019.
32 Toby Hill. Blackrock goes green? investment giant joins climate action, 2020.
33 Nicolas Houy. Rational mining limits bitcoin emissions. *Nature Climate Change*, 2019.
34 G Huberman et al. Monopoly without a monopolist: An economic analysis of the bitcoin payment system. *Social Science Research Network*, 2017.
35 Digital Currency Initiative. 51% attacks, 2020. URL: `https://dci.mit.edu/51-attacks`.

36 Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In *Secure information networks*, pages 258–272. Springer, 1999.

37 Ari Juels. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *NDSS*, 1999.

38 Ghassan Karame et al. Double-spending fast payments in bitcoin. In *CCS*, 2012.

39 Aggelos Kiayias et al. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Crypto*, 2017.

40 Lucianna Kiffer et al. A better method to analyze blockchain consistency. In *CCS*, 2018.

41 Eleftherios Kokoris Kogias et al. Enhancing bitcoin security and performance with strong consistency via collective signing. In *USENIX*, 2016.

42 Joshua A Kroll et al. The economics of Bitcoin mining or, Bitcoin in the presence of adversaries. In *WEIS*, 2013.

43 Ben Laurie and Richard Clayton. Proof-of-work proves not to work. In *WEIS*, 2004.

44 K. Liao and J. Katz. Incentivizing blockchain forks via whale transactions. In *FC*, 2017.

45 Michael Lynn. Scarcity effects on value: A quantitative review of the commodity theory literature. *Psychology & Marketing*, 1991.

46 Patrick McCorry et al. Smart contracts for bribing miners. In *FC*, 2018.

47 Andrew Miller et al. Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions. In *CCS*, 2015.

48 Michael Mirkin et al. Bdos: Blockchain denial-of-service. In *CCS*, 2020.

49 Camilo Mora et al. Bitcoin emissions alone could push global warming above 2 c. *Nature Climate Change*, 2018.

50 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

51 Arvind Narayanan et al. *Bitcoin and cryptocurrency technologies: a comprehensive introduction.* Princeton, 2016.

52 Kartik Nayak et al. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *IEEE European SP*, 2016.

53 T. Neudecker and H. Hartenstein. Short paper: An empirical analysis of blockchain forks in bitcoin. In *FC*, 2019.

54 BBC news. Tesla will no longer accept bitcoin over climate concerns, says musk, 2021.

55 Shunya Noda et al. An economic analysis of difficulty adjustment algorithms in proof-of-work blockchain systems. In *EC*, 2020.

56 Andrew Poelstra et al. Distributed consensus from proof of stake is impossible. *Self-published Paper*, 2014.

57 Jamie Redman. Bitcoin gold 51% attacked - network loses $70,000 in double spends, 2020.

58 Tim Roughgarden. Transaction fee mechanism design. *arXiv*, 2021.

59 Ayelet Sapirshtein et al. Optimal selfish mining strategies in Bitcoin. In *FC*, 2016.

60 Jon Truby. Decarbonizing bitcoin: Law and policy choices for reducing the energy consumption of blockchain technologies and digital currencies. *Energy research & social science*, 2018.

61 Itay Tsabary et al. Mad-htlc: Because htlc is crazy-cheap to attack. *S&P*, 2021.

62 Itay Tsabary and Ittay Eyal. The gap game. In *CCS*, 2018.

63 Itay Tsabary, Alexander Spiegelman, and Ittay Eyal. Tuning pow with hybrid expenditure – extended version. *arXiv preprint*, 2021. `arXiv:1911.04124`.

64 Aviv Yaish and Aviv Zohar. Pricing asics for cryptocurrency mining. *arXiv*, 2020. `arXiv:2002.11064`.

On Cryptocurrency Wallet Design

Ittay Eyal
Technion, Haifa, Israel
IC3

Abstract

The security of cryptocurrency and decentralized blockchain-maintained assets relies on their owners safeguarding secrets, typically cryptographic keys. This applies equally to individuals keeping daily-spending amounts and to large asset management companies. Loss of keys and attackers gaining control of keys resulted in numerous losses of funds.

The security of individual keys was widely studied with practical solutions available, from mnemonic phrases to dedicated hardware. There are also techniques for securing funds by requiring combinations of multiple keys. However, to the best of our knowledge, a crucial question was never addressed: How is wallet security affected by the number of keys, their types, and how they are combined? This is the focus of this work.

We present a model where each key has certain probabilities for being *safe*, *lost*, *leaked*, or *stolen* (available only to an attacker). The number of possible wallets for a given number of keys is the Dedekind number, prohibiting an exhaustive search with many keys. Nonetheless, we bound optimal-wallet failure probabilities with an evolutionary algorithm.

We evaluate the security (complement of failure probability) of wallets based on the number and types of keys used. Our analysis covers a wide range of settings and reveals several surprises. The failure probability general trend drops exponentially with the number of keys, but has a strong dependency on its parity. In many cases, but not always, heterogeneous keys (not all with the same fault probabilities) allow for superior wallets than homogeneous keys. Nonetheless, in the case of 3 keys, the common practice of requiring any pair is optimal in many settings.

Our formulation of the problem and initial results reveal several open questions, from user studies of key fault probabilities to finding optimal wallets with very large numbers of keys. But they also have an immediate practical outcome, informing cryptocurrency users on optimal wallet design.

2012 ACM Subject Classification Security and privacy → Formal security models

Keywords and phrases cryptocurrency, wallet, key-management, authentication

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2021.4

Related Version *Previous Version*: `https://eprint.iacr.org/2021/1522`

Funding The author is grateful for the support of ISF grant 1641/18 and BSF.

1 Introduction

Cryptocurrency systems maintain their state in a database called a blockchain. The state includes user's data as well as their token balances. To change their data, and in particular to order transactions of their funds, users append data structures called *transactions* to the blockchain. To authenticate themselves, users include in the transaction a proof of their (typically pseudonymous) identity.

Unlike common centralized systems, the choice of the authentication method is at the hands of the users themselves. A user defines, with a blockchain transaction, a predicate for authentication to access her digital assets. Transaction that access those assets must include inputs to make this predicate true. The predicate is called a *wallet*, and it is specified in a dedicated programming language as a *smart contract*. Typically, wallets require one or more cryptographic signatures. The *security* of a wallet, its probability of not failing, thus relies on the safekeeping of one or more secrets – private keys maintained by the user.

In classical systems like bank accounts and credit cards, accounts are identified and users are insured against theft. They are allowed to revoke and refresh their keys using alternative authentication techniques [13]. None of those mechanisms apply to cryptocurrencies, and if a user's keys are lost or leaked, she immediately loses access to her funds. The problem applies equally to personal wallets, holding small amounts for daily use, and to large actors such as companies holding their own[1] or their clients'[2] funds.

Previous work (§2) proposed techniques to maintain keys [5, 11], including locally-stored files, possibly encrypted, dedicated hardware [1], hosted online services, paper wallets, and brain wallets (i.e., committing the key to memory). Other work discovered techniques to implement threshold signatures [19, 15, 14], where multiple secrets are necessary to produce a signature but, unlike classical secret sharing [21], the private key is never learnt by any party. Password-protected secret sharing [2, 17] allows users to store a secret shared between multiple servers, overcoming server faults, and without the servers learning the secret without a password. Companies[3] and open-source projects[4] use wallets that require two keys, or alternatively one-out-of-two keys, two-out-of-three keys, or more elaborate schemes.

Nonetheless, to the best of our knowledge, the question of how many keys a user should maintain and what combination is the most secure was never formally explored. This is the goal of this work.

We model the system (§3) by considering a principal, called an *owner*, that wishes to secure cryptocurrency tokens. The owner can store one or more secrets called *keys*. These could be cryptographic signing keys, portions thereof, or any other secrets. If only the owner can access a key we call that key *safe*. But keys can suffer three types of *faults*. First, they can be *lost*; e.g., in a discarded hard-drive[5], a forgotten password[6], due to a fire, etc. Secondly they can be *leaked* to the adversary, e.g., by gaining access to a machine, keystroke logging, or guessing [22, 6]. Finally, they can be *stolen*; this is a combination of the previous faults, where the adversary gains access to the key and the owner loses access to it. This could happen, e.g., if the key is controlled by an online service that is compromised, or if an attacker physically steals a written secret or even a hardware key [16]. There is a certain probability for each of the keys to suffer each of the faults. Note that the four states cannot be reduced: the probability of loss and leakage can be 2% each with a theft probability of either 0 or 1%, for example. For simplicity we assume that fault probabilities of the keys are independent of one another. We also assume that the owner knows these probabilities.

To secure her funds, the owner locks her tokens in the cryptocurrency system with a smart contract that can require an arbitrary combination of the keys. For two keys it can require either key, so if one is lost she can use the other. She could also require both keys, so if one is leaked the adversary cannot move the tokens. We call the set of possible combinations defined in the contract a *wallet*. If a party has the keys for one of the combinations we say she can *satisfy* the wallet. The wallet *fails* if the owner cannot satisfy it or if an adversary can. The former happens if enough keys are either lost of stolen, and the latter if enough keys are either leaked or stolen. The owner should choose the *best wallet* (lowest failure probability) given the fault probabilities of the keys.

[1] `https://edition.cnn.com/2021/02/08/investing/tesla-bitcoins/index.html`
[2] `https://www.forbes.com/advisor/investing/best-crypto-exchanges`, `https://www.cmegroup.com/markets/cryptocurrencies.html`
[3] `https://keys.casa/`, `https://zengo.com/`
[4] `https://electrum.readthedocs.io/en/latest/multisig.html`
[5] `https://www.bbc.com/news/uk-wales-south-east-wales-25134289`
[6] `https://www.nytimes.com/2019/02/05/business/quadriga-cx-gerald-cotten.html`

To analyze wallet design we use two techniques (§4). First, for wallets with up to 5 keys we are able to complete an exhaustive search of the wallet space. By enumerating the probabilities of all key states and calculating the failure probability of each wallet we can find the optimal wallet given key fault probabilities.

The wallet space grows super-exponentially with the number of keys prohibiting an exhaustive search for large wallets. However, we can bound the failure probabilities of the optimal wallets using evolutionary optimization [10]. Roughly, we construct a population of wallet candidates and iteratively perturb them and select the best for the next generation.

We begin our analysis (§5) with a detailed review of the optimal wallets for up to 4 keys with specific fault probabilities. Even with two keys, we find that there are prominent cases where the owner is better off by using only one key and ignoring the other. This happens with homogeneous keys (that have the same probabilities) if the theft probability is non-negligible and the loss and leak probabilities are similar. With three keys, we confirm that the common 2/3 approach [12] is indeed the optimum for a variety of realistic settings. We illustrate why different optimal wallets are better in different settings by considering the probabilities that the owner and that the attacker can satisfy a wallet. With four keys, two distinct natural extrapolations of the 2/3 solution are optimal for a wide range of probabilities, namely either all pairs or all triads. However, in prominent cases, where theft probability is non-negligible and loss and leak probabilities are similar, the optimum is a different, asymmetric wallet.

The owner can choose how many keys to use for its wallet. Our next step is therefore to evaluate the effect of the number of keys of wallet security. As expected, the optimal-wallet failure probability drops in a general exponential trend with the number of keys. However, when the theft probability is non-negligible, there is a strong dependence on parity. For example, when all keys have the same theft probability but no other faults, failure probability does not improve when adding a key to an odd-key wallet.

Finally, the owner has a choice of which keys to use for its wallet. For example, she could use a key stored in a mobile-device and two hardware keys, or vice versa. By considering the design space for 3-key wallets, we find that it is often, but not always, best to have heterogeneous keys.

We conclude (§6) by reviewing several open directions exposed by this work. These include user studies to quantify the probabilities of key faults, dependency among key fault probabilities, and techniques to identify optimal wallets with very large numbers of keys. However, the results presented here are of immediate importance to users, who can estimate their keys' fault probabilities, choose keys wisely, and thus improve the security of their funds. We intend to open-source the tools presented here.

2 Related Work

No work we are aware of analyzes cryptocurrency wallet design, though many works discuss tools for wallet design and implement a variety of designs.

Kirstein et al. [18] assume users have two storage tiers, one that is harder to access but is more secure, and another that is easy to access but is less secure. They design a wallet contract based on Moser et al. [20] that takes advantage of the two tiers to provide both good security and good accessibility. Earlier work by Baratam [3] uses a similar technique to combine a variety of keys with different properties. He combines user-maintained keys with keys hosted by remote services and proposes a wallet contract to minimize failure probability. Both of the proposals focus on the mechanics of the wallets, but neither of them describes the key fault probabilities, analyzes the resultant wallet failure probabilities, or addresses the choice of type and number of keys.

Eskandari et al. [11] provide a taxonomy of individual key storage options, including local storage, encrypted local storage, offline storage, hosted, etc. Evaluated criteria cover security, as well as usability and deployability. However, they do not consider multi-key wallets and overall wallet failures, only the security of individual keys.

Bonneau et al. [5] devote a whole chapter in their Systemization of Knowledge work to client-side security. In addition to per-key storage options, they also discuss the mechanics for implementing a symmetric k/n wallet, but not a formal treatment of its superior wallet security.

In their review of cryptocurrency research challenges, Barber et al. [4] also devote a chapter to client-side security. But they, too, focus on the properties of individual keys. They explicitly discuss the threat of benign key loss, but do not consider theft, and do not quantify fault probabilities and their effect.

3 Model

We explain the reasoning and simplifications resulting in our model (§3.1) followed by the formal specification (§3.2).

3.1 Rationale

A cryptocurrency user secures her assets by defining arbitrary logic to authorize access to her assets. We call this logic a *wallet*. It is implemented in a so-called smart-contract for the relevant blockchain. Once the wallet is implemented in a smart contract, assets can be allocated (sent) to it on the blockchain. When issuing commands that affect those assets, the user publishes a transaction with inputs to the smart contract, and the transaction takes effect only if the contract authorizes it.

The transaction is first published via a peer-to-peer network, and subsequently placed in the blockchain by one of its operators, called *miners*. This mechanism means that the authentication mechanism should prohibit malicious parties from being able to authenticate themselves based on observed transactions. Otherwise, miners (or really anyone) [8] could take advantage and relieve the user of her assets: They would observe a transaction, replace the order with another, and falsely authenticate.

Therefore, wallets are typically implementing by requiring cryptographic signatures matching predefined public keys. A wallet can combine any number of signature requirements, e.g., for two particular public keys, for any pair out of three options, etc. Moreover, the contract can require only a single signature, but the user partitions her secret into several parts and combines them offline before issuing a transaction. Either way, the result is the same – the wallet is secured by a set of secrets we call keys (even if they are parts of a single cryptographic key) requiring a combination of the keys to authenticate. The wallet is thus a predicate of this set of keys, and if a party has enough keys to make this predicate true we say she can *satisfy* the wallet.

We say a wallet fails if the user loses control of her assets, which could happen due to two reasons. First, the owner might lose access to keys and not be able to satisfy the wallet. Secondly, an adversary could gain access to a combination of keys that allows her to satisfy the wallet and steal the assets. This second option includes scenarios where both parties can satisfy the wallet. In practice, in such scenarios the parties can enter a bidding war, each paying more fees, trying to get the miners to place her transaction in the blockchain. The fees can be arbitrarily high, with the attacker possibly willing to pay almost the entire wallet amount to make a profit. We therefore consider the wallet failed in such scenarios.

An owner can add offline security measures to her keys. For example, she could encrypt locally-stored keys or require a PIN access code to a hardware signing device. These approaches simply reduce the probability that the attacker gains access to a key but also increase the probability that the owner loses access [5]. They are thus covered by our model.

An owner might lose access to her keys gradually. For example, she forgets one memorized key, and after several months loses another paper key due to a basement flood. An adversary can also gain key access gradually. For example, a hacker steals a hardware signing device and subsequently retrieves a typed key with a keystroke logger as the user tries to save her assets.

In our model we flatten such series of events of keys being lost and leaked to probabilities per key. One can think of those as being evaluated at the time at which the owner wishes to transact her funds. At this point, there is a certain probability that the adversary had gained access to the keys, or that the owner had lost access.

3.2 Model Details

An *owner* maintains a set of n keys $k_1, \dots, k_n$. Each key is in one of four states:
safe Only the user has access,
loss No one has access,
leak Both the user and the adversary have access, or
theft Only the adversary has access,
denoted $\mathcal{S}$, i.e., for each $1 \le i \le n$: $k_i \in \mathcal{S} = \{\mathsf{safe}, \mathsf{loss}, \mathsf{leak}, \mathsf{theft}\}$.

A scenario σ is the state vector of each of the keys, and denote by Σ_n the set of all scenarios with n keys, i.e., $\sigma \in \Sigma_n = (\mathsf{safe}, \mathsf{loss}, \mathsf{leak}, \mathsf{theft})^n$. Denote by σ_i the state of key i in scenario σ.

Denote by σ^O and σ^A the binary *availability vectors* of each of the keys in scenario σ for the owner and for the adversary, respectively. For the state of key i, σ_i, the availabilities are σ_i^O and σ_i^A. For example, if $\sigma_i = \mathsf{safe}$ then $\sigma_i^O = \mathit{True}$ and $\sigma_i^A = \mathit{False}$. Table 1 summarizes the translation from state to availability.

The states of the different keys are determined by a probability space specified by independent probabilities of each of the keys. These probabilities are described by a tuple P of probability vectors P^{safe}, P^{loss}, P^{leak} and P^{theft}. The probabilities that key i is in each of the states are denoted P_i^{safe}, P_i^{loss}, P_i^{leak}, and P_i^{theft} for safe, loss, leak, and theft, respectively. The sum of probabilities for each key i is one, $P_i^{safe} + P_i^{loss} + P_i^{leak} + P_i^{theft} = 1$.

The probability of a scenario can be calculated given the probability vector tuple. For example, for three keys, the probability that the first is safe, the second is lost and the third is leaked is $P_1^{safe} \times P_2^{loss} \times P_3^{leak}$. In general: ($\mathbb{1}_c$ is the indicator function, it equals 1 if the condition c holds and 0 otherwise)

$$\Pr[\sigma] = \prod_{i=1}^{n} \left(\sum_{s \in \mathcal{S}} \mathbb{1}_{\sigma_i = s} \times P_i^s \right) . \tag{1}$$

With n keys there are 4^n possible states, each key being in one of its four possible states. We can calculate the probability of each scenario and derive the key availability probabilities for each party. Table 2 shows 5 scenarios from the $4^2 = 16$ possible with 2 keys.

A wallet w is a predicate of the availability of the n keys. For example, a wallet w that requires the availability of either keys k_1 and k_2 or keys k_2 and k_3 is defined by $w(a_1, a_2, a_3) \triangleq (a_1 \wedge a_2) \vee (a_2 \wedge a_3)$. By slight abuse of notation, for a vector $v = (v_1, v_2, v_3)$ we write $w(v)$ for $w(v_1, v_2, v_3)$.

Table 1 Key state and availability.

σ_i	${\sigma^O}_i$	${\sigma^A}_i$
safe	*True*	*False*
loss	*False*	*False*
leak	*True*	*True*
theft	*False*	*True*

Table 2 State enumeration.

σ_1	σ_2	probability	σ^O	σ^A
safe	safe	$(P^{safe})^2$	$(True, True)$	$(False, False)$
safe	loss	$P^{safe} \times P^{loss}$	$(True, False)$	$(False, False)$
safe	leak	$P^{safe} \times P^{leak}$	$(True, True)$	$(False, True)$
safe	theft	$P^{safe} \times P^{theft}$	$(True, False)$	$(False, True)$
loss	safe	$P^{loss} \times P^{safe}$	$(False, True)$	$(False, False)$
...	...	...	...	...

We denote by $w^{\mathsf{k/n}}$ the wallet that is satisfied by any set of k out of the n keys. The wallet w_n^{AND} is the wallet requiring all n keys ($w^{\mathsf{n/n}}$), and the wallet w_n^{OR} is the wallet requiring any of the n keys ($w^{\mathsf{1/n}}$).

A wallet w is *successful* in a scenario σ if and only if the adversary cannot satisfy it and the owner can, i.e., $w(\sigma^O) \wedge \neg w(\sigma^A)$. Otherwise, the wallet has *failed* in this scenario.

Given a probability vector tuple P, we can calculate for wallet w its success and failure probabilities, $p^P_{success}(w)$ and $p^P_{failure}(w)$, respectively. For success, we sum the probabilities of all states where the wallet is successful,

$$p^P_{success}(w) = \sum_{\sigma \in \Sigma_n} \Pr[\sigma] \times \mathbb{1}_{w(\sigma^O)} \times \mathbb{1}_{\neg w(\sigma^A)} \ , \tag{2}$$

and the failure probability is the complement $p^P_{failure}(w) = 1 - p^P_{success}(w)$.

We denote the fact a wallet w is better than wallet w', i.e., $p^P_{failure}(w) < p^P_{failure}(w')$ by $w' \prec_P w$, omitting the subscript ($w' \prec w$) when P is clear from the context.

4 Methodology

To find the optimal wallet for a given probability vector tuple, we search the entire design space when possible (§4.1) or use an approximation (§4.2) otherwise.

4.1 Exhaustive Search

For small numbers of keys we can find the optimal wallet by exhaustively calculating the failure probabilities of all possible wallets. We observe that each wallet is a monotone boolean function – if a party can satisfy the wallet, being able to access another key will not prohibit it from satisfying the wallet. The exact number of such functions, the Dedekind number [9] minus 2 (excluding the constant functions True and False), is only known up to 8 keys [7, 23, 24]. It grows super-exponentially, making complete coverage of all possible wallets intractable. Complete coverage also becomes intractable due to the exponentially growing number of key states (4^n).

To enumerate all wallets, we express wallets as a logical *or* of sets of logical *and*'s. We call each set of *and*-ed keys a *combination*. For example, for two keys we consider four possible wallets, requiring one combination of both keys ($k_1 \wedge k_2$), requiring either of them $(k_1) \vee (k_2)$ (two combinations), but also the asymmetric options of requiring one (k_1) or the other (k_2).

We denote by $c \subset c'$ the fact that all keys in combination c are also in combination c' and say c' *covers* c. We exclude wallets where one combination covers another, e.g., $(k_1 \wedge k_2 \wedge k_3) \vee (k_1 \wedge k_2)$ as it is the same function as $(k_1 \wedge k_2)$, which is evaluated.

There are a total of $(2^n - 1)$ possible non-zero combinations, and we consider all sets of those combinations that are not redundant due to one combination covering another.

Algorithm 1 Wallet enumeration.

```
function enumerateWallets ($baseWallet, prevCombi, keyCount$)
    $wallets \leftarrow \emptyset$
    for $combi = next(prevCombi)$ to $lastCombi(n)$ do
        if $\nexists c \in baseWallet : c \subset combi$ then          (Skip redundant combinations)
            $currWallet \leftarrow baseWallet \cup \{combi\}$
            $wallets \leftarrow wallets \cup \{currWallet\}$                (With new combination)
            $wallets \leftarrow wallets \cup$ enumerateWallets($currWallet, combi, keyCount$)
                                                          (With new combination and others)
    return($wallets$)
```

For example, with three keys there are 18 possible wallets, as follows,

$$(k_1), (k_2), (k_3), \tag{3a}$$
$$(k_1) \vee (k_2), (k_1) \vee (k_3), (k_2) \vee (k_3), \tag{3b}$$
$$(k_1) \vee (k_2) \vee (k_3), \tag{3c}$$
$$(k_1 \wedge k_2), (k_1 \wedge k_3), (k_2 \wedge k_3), \tag{3d}$$
$$(k_1 \wedge k_2) \vee (k_3), (k_1) \vee (k_2 \wedge k_3), (k_2) \vee (k_1 \wedge k_3), \tag{3e}$$
$$(k_1 \wedge k_2) \vee (k_1 \wedge k_3), (k_1 \wedge k_2) \vee (k_2 \wedge k_3), (k_1 \wedge k_3) \vee (k_2 \wedge k_3), \tag{3f}$$
$$(k_1 \wedge k_2) \vee (k_1 \wedge k_3) \vee (k_2 \wedge k_3), \tag{3g}$$
$$(k_1 \wedge k_2 \wedge k_3) \ . \tag{3h}$$

We enumerate the wallets recursively, as shown in Algorithm 1. The algorithm utilizes a lexicographic order of the combinations, implemented with the function *next* that returns the next combination in order and *last* that return the last combination, namely $k_1, \dots, k_n$. We call the algorithm with $baseWallet = \emptyset$, $prevCombi = 0$, and the desired number of keys, n. The recursive algorithm returns the set of all wallets with all non-redundant subsequent combinations, considering redundancy with wallets in *baseWallet*.

4.2 Evolutionary Approximation

To analyze larger numbers of keys, beyond the reach of an exhaustive search, we search the space with an evolutionary algorithm [10] (Algorithm 2). We bootstrap the algorithm with a random *population* – a multiset of n_{pop} wallets. The algorithm then iteratively improves this population. In each iteration i we produce a new generation based on the population of iteration $i-1$.

We first select a multiset of n_{pop} wallets as follows. For each selection we consider $n_{selection}$ wallets uniformly at random (UAR) from iteration $i-1$ and select the best (lowest failure rate) among them. Good wallets are likely to appear multiple times in the new multiset.

We then perturb each wallet in the selection by choosing one of its combinations uniformly at random and flipping each of its key requirements with independent probability $p_{perturb}$. For example, for a three-key wallet combination of $k_1 \wedge k_2$, key k_1 is removed with probability $p_{perturb}$, key k_2 is removed with probability $p_{perturb}$, and key k_3 is added with probability $p_{perturb}$.

The resultant wallet multiset is then taken as the population of the next iteration. We repeat this until there is no improvement in wallet failure probability for $n_{stability}$ generations.

Algorithm 2 Evolutionary Wallet Optimization.

```
input : $n, P; n_{pop}, n_{selection}, p_{perturb}, n_{stability}$
pop ← $n_{pop}$ random wallets
$\delta \leftarrow 0$                                              (Convergence counter)
$best \leftarrow 0$                                            (Best wallet's probability)
while $\delta < n_{stability}$ do                           (While not stable for long enough)
    if $\max_w \{p^P_{success}(w) | w \in pop\} > best$ then
        $best \leftarrow \max_w \{p^P_{success}(w) | w \in pop\}$
        $\delta \leftarrow 0$                                      (Reset convergence counter)
    else
        $\delta \leftarrow \delta + 1$
    $selected \leftarrow \{\{\arg\max_w \{p^P_{success}(w) | w \in pop\}\}\}$            (Keep best wallet)
    while $|selected| < n_{pop}$ do
        $W \leftarrow$ choose $n_{selection}$ UAR from pop
        $selected \leftarrow selected + \{\{\arg\max_w \{p^P_{success}(w) | w \in W\}\}\}$
    $nextGen \leftarrow \{\{\}\}$
    foreach wallet $w$ in $selected$ do
        $c \overset{\$}{\leftarrow} w \cup \{\emptyset\}$          (Choose UAR a combination in $w$ or an empty combination)
        $c' \leftarrow \emptyset$
        foreach $i \in [n+1]$ do                  (With probability $p_{perturb}$ toggle each key)
            with probability $p_{perturb}$
                $c' \leftarrow c' \cup \begin{cases} \{k_i\} & k_i \notin c \\ \emptyset & k_i \in c \end{cases}$
            else
                $c' \leftarrow c' \cup \begin{cases} \{k_i\} & k_i \in c \\ \emptyset & k_i \notin c \end{cases}$
        $w' \leftarrow w \setminus \{c\} \cup \{c'\}$                  (Replace perturbed combination)
        $nextGen \leftarrow nextGen + \{w'\}$              (Replace perturbed wallet)
    $pop \leftarrow nextGen$
```

5 Optimal Wallets

We are now ready to analyze the optimal wallet choices and the resultant probabilities. We begin (§5.1) by studying in detail optimal wallets with small numbers homogeneous keys. Next (§5.2) we look at the effect of the number of keys on wallet failure probability. Finally (§5.3) we turn our attention to optimal wallets for heterogeneous keys.

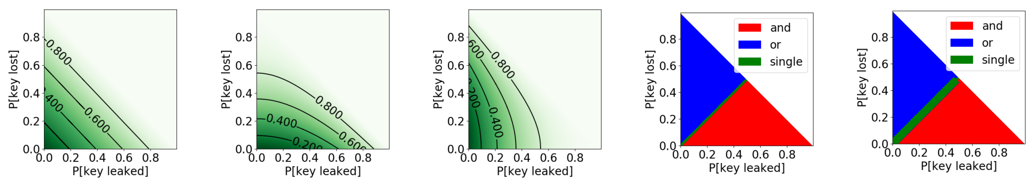
5.1 Homogeneous Keys

For homogeneous keys we denote state probabilities by P^{safe}, P^{loss}, P^{leak}, and P^{theft}, omitting the key indices.

5.1.1 One Key

For a wallet with a single key, $w^{\mathsf{Single}}(k_1) = k_1$, the probability for success is simply the probability that the key was neither lost, leaked, nor stolen, i.e.,

$$p_{success}(w^{\mathsf{Single}}) = P^{safe} \quad . \tag{4}$$



(a) Failure prob. Single key. **(b)** Failure prob. $k_1 \wedge k_2$. **(c)** Failure prob. $k_1 \vee k_2$. **(d)** Best wallet. **(e)** Best wallet; $P^{theft} = 0.04$.

Figure 1 Two-key wallets; $P^{theft} = 0.01$ unless noted otherwise.

Figure 1a illustrates the success probability of the wallet for different values of P^{leak} and P^{loss} with a constant $P^{theft} = 0.01$. The isolines show equal failure probability, indicating the lowest failure probability reaches 0.01 when $P^{leak} = P^{loss} = 0$.

5.1.2 Two keys

With two keys several wallets are possible:

AND Require both keys (Figure 1b). For success we require that either both keys are safe, or one is safe and the other is leaked. If either key is lost or stolen then the wallet fails as the owner cannot satisfy it.

$$p_{success}(w_2^{\mathsf{AND}}) = (P^{safe})^2 + 2P^{safe}P^{leak} \ . \tag{5}$$

OR Require at least one of the keys (Figure 1c). For success we require that either both keys are safe or one is safe and the other is lost. If either key is leaked or stolen then the wallet has failed.

$$p_{success}(w_2^{\mathsf{OR}}) = (P^{safe})^2 + 2P^{safe}P^{loss} \ . \tag{6}$$

The symmetry between these two wallets is clear, and it is evident that when $P^{leak} < P^{loss}$ the OR wallet is superior, and vice versa. But there is a third option, which simply ignores one of the keys; since the probabilities are the same for both keys it doesn't matter which one:

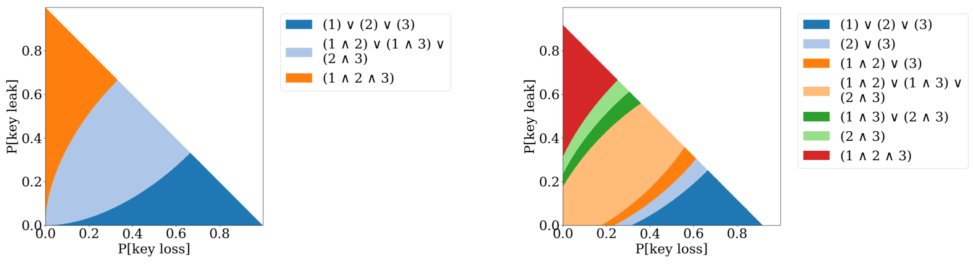
Single Require that the key is neither lost nor leaked (Figure 1a, Equation 4).

Comparing these probabilities shows that there is a region where using a single key is more secure than either of the other options – when the loss and leak probabilities are similar. To be better than the OR wallet, it should hold that $(P^{safe})^2 + 2P^{safe}P^{loss} < P^{safe}$, and similarly for the AND wallet, i.e.,

$$P^{loss} - P^{leak} < P^{theft} \Rightarrow w_2^{\mathsf{OR}} \prec w^{\text{single}}, \tag{7a}$$
$$P^{leak} - P^{loss} < P^{theft} \Rightarrow w_2^{\mathsf{AND}} \prec w^{\text{single}}, \tag{7b}$$

so if $|P^{loss} - P^{leak}| < P^{theft}$, the optimal 2-key wallet is w^{Single}.

Figures 1d-1e show which wallet dominates each region in the settings space with theft probabilities of 0.01 and 0.04. Note that although we draw the full range (fault probabilities almost up to 1), the practical range is where fault probabilities are small, and the single-key wallet is optimal in a significant portion of this region.



(a) $P^{theft} = 0.0$.   **(b)** $P^{theft} = 0.08$.

Figure 2 Three-key optimal wallet for varying theft probability.

5.1.3 Three keys

With three keys there are 18 possible wallets (§4.1). Figure 2 shows the optimal wallets in a range of settings. In such figures we write i instead of k_i to reduce clutter.

We see that the common wisdom in wallet design [12] is correct in a wide range of settings: The wallet $w^{2/3} = (k_1 \wedge k_2) \vee (k_2 \wedge k_3) \vee (k_3 \wedge k_1)$ is optimal when P^{loss} and P^{leak} are roughly the same. As expected, the OR wallet $(k_1 \vee k_2 \vee k_3)$ is optimal when the loss probability is much higher than the leak probability, and the AND wallet is optimal when a leak is much more likely than a loss.

When there is no theft (Figure 2a), symmetric wallets are optimal across the range of loss and leak probabilities. However, there are regions in the homogeneous settings space where asymmetric wallets are optimal, using only a pair of the keys, or all three, e.g., $(k_1 \wedge k_2) \vee (k_3)$. This occurs when theft probability is non-negligible, and becomes pronounced when it is large, as shown for $P^{theft} = 0.8$ in Figure 2b. Nonetheless, the common $w^{2/3}$ wallet remains optimal in the key portion of the space where the fault probabilities are small.

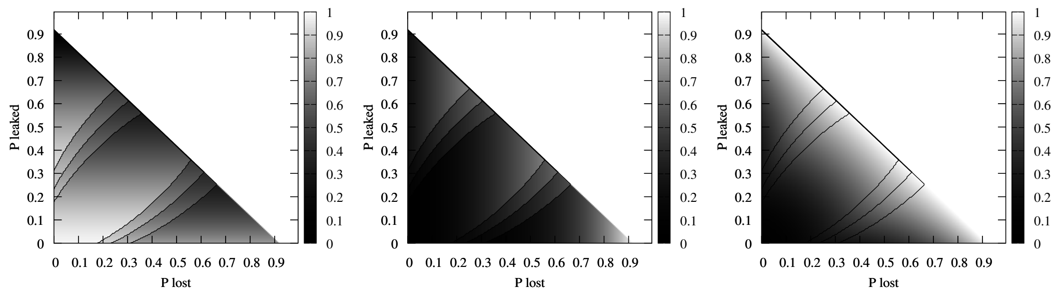
Owner and Adversary Perspectives

To better understand the wallet changes, we explore the probabilities that each of the owner and adversary can satisfy the optimal wallet. A wallet does not fail if the owner has the keys and the adversary does not, and this is the probability we aim to maximize. Figure 3c shows the failure probability of the optimal wallet with $P^{theft} = 0.08$ and varying values of P^{loss} and P^{leak}. The black lines show the transition lines between different optimal wallets (cf. Figure 2). Figures 3a and 3b show the probability that the owner and the adversary (respectively) can satisfy the optimal wallet.

We observe that the wallet success probability is continuous as expected – a switch from one wallet to another occurs at the point where their success probabilities are the same. Figure 3a shows where the owner's decreasing probability leads to a wallet switch, shown as a darkening color switching to light across a boundary. Figure 3b shows the contrary, where the adversary's increasing probability leads to a wallet switch, shown as a lightning color switching to dark across a boundary.

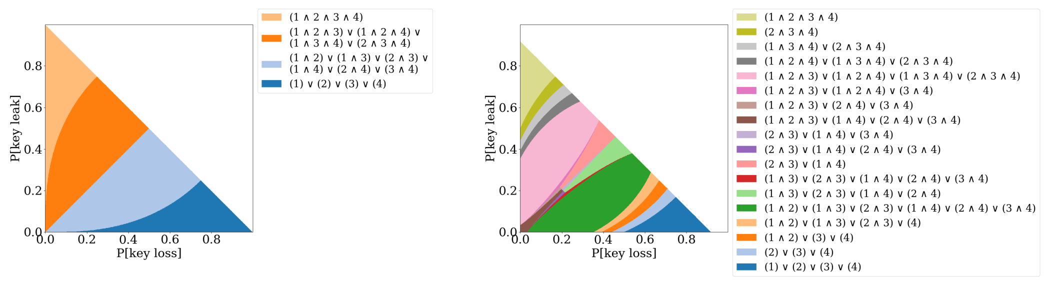
5.1.4 Four Keys

When moving to four keys, it is natural to consider an optimal wallet based on the three-key case, $w^{2/3}$. But extrapolating to four keys we could require either any pair, $w^{2/4}$, or any triad, $w^{3/4}$. If there is no theft, Figure 4a shows that indeed these two options dominate a



(a) Owner access probability. **(b)** Adversary access probability. **(c)** Wallet failure probability.

Figure 3 Wallet failure breakdown. $P^{theft} = 0.08$.



(a) $P^{theft} = 0.0$. **(b)** $P^{theft} = 0.08$.

Figure 4 Best four-key wallets without theft.

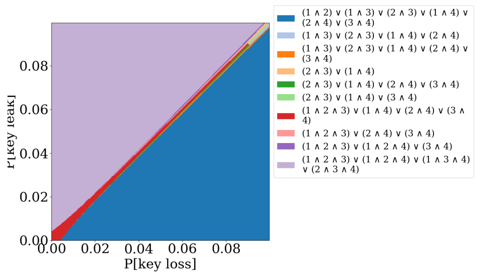
wide range of settings. Specifically, if P^{loss} is larger than P^{leak}, then pairs is the right choice, as less keys are required from the owner. And if P^{leak} is larger than P^{loss} then triads is the right choice, as more keys are required from the attacker. As expected, the OR wallet is optimal when loss is very likely and leak isn't; and the AND wallet is optimal when leak is very likely but loss isn't.

When P^{theft} is positive, as in the two-key case, asymmetric solutions become optimal in a wider range of cases. Figure 4b shows the case for theft probability of 0.08 (taking a large value to illustrate the effect). The two extrapolations of the 3-key case remain dominant in many settings. But now, unless P^{loss} and P^{leak} are significantly different, the optimal wallets are asymmetric.

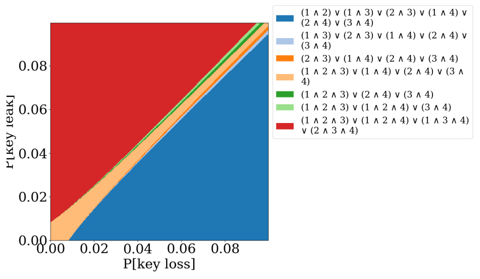
Figure 5 focuses on the range of smaller fault probabilities, with loss and leak of up to 0.10, and with more realistic theft probabilities of 0.01 and 0.02. We see that in this region, asymmetric wallets become dominant in the most practical region. When both P^{loss} and P^{leak} are roughly similar and below 0.01, the optimal wallet is of the type $(1 \wedge 2 \wedge 3) \vee (1 \wedge 4) \vee (2 \wedge 4) \vee (3 \wedge 4)$ (or any permutation, as the key probabilities are identical).

5.2 Number of keys

The number of keys, has major effect on wallet security. While 2-3 keys might be all that is necessary or technically feasible for an individual, much larger numbers could be practical when storing large amounts, e.g., by financial institutions. In such cases, a group of 6 executives, maybe more, can be assigned to keep 6 different keys.

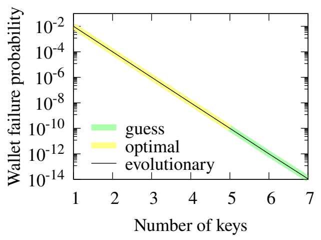


(a) $P^{theft} = 0.01$.

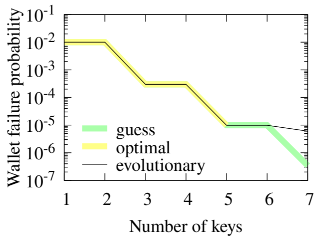


(b) $P^{theft} = 0.02$.

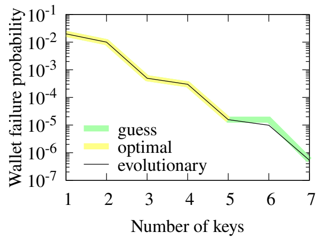
Figure 5 Best four-key wallets without theft – small probabilities.



(a) $P^{loss} = 0.01$, $P^{leak} = 0.01$.



(b) $P^{theft} = 0.01$.



(c) First: $P^{loss} = P^{leak} = 0.01$; Next: $P^{theft} = 0.01$.

Figure 6 Wallets with many keys and one fault type.

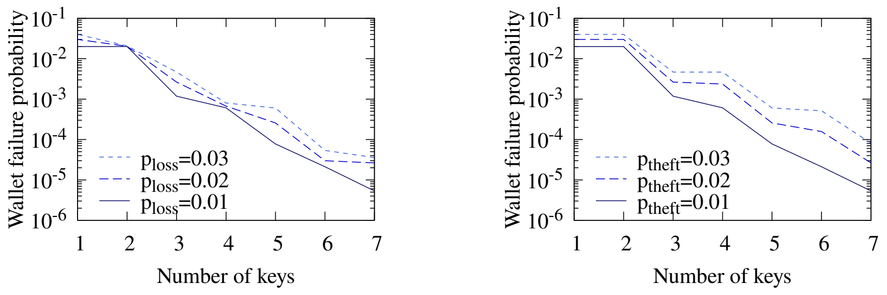
We assume all keys have the same fault probabilities and vary the number of keys from 1 to 7. If the only fault is loss with some probability P^{loss}, the optimal wallet is w_n^{OR}, and its failure probability drops exponentially with the number of keys (Figure 6a). A similar result is achieved with the w_n^{AND} wallet if only leaks are possible.

In a theft-only setting the result is different (Figure 6b). While the overall trend remains exponential, the improvement only emerges on odd numbers of keys. With an odd n number, the best solution is the symmetric $n-1$ optimal solution. The figures show the optimal wallet failure probability, where it can be calculated; with larger numbers of keys they show the failure probability of the best-guess-wallet. The guesses include all symmetric wallets with m keys where $m \leq n$, i.e., all symmetric wallets for smaller key sets.

In all cases the evolutionary algorithm closely approximates the optima when starting from a random population, demonstrating its effectiveness. In the next evaluations we bootstrap with the best symmetric key available, so it can only improve further.

In practice, increasing the number of keys could imply using keys with individual larger fault probabilities. For example, by assigning key keeping responsibilities to more, less trusted, individuals. Figure 6c shows the security when the first wallet can be lost or leaked, but not stolen ($P^{loss} = P^{leak} = 0.01$), and the rest of the keys can only be stolen ($P^{theft} = 0.01$).

Figure 7 shows the approximate optimal wallet failure probability with wallets that suffer only loss and theft with different fault probabilities and different numbers of keys. The exponential improvement with key number results, for example, in an order-of-magnitude improvement when theft (loss) probability is a constant 0.01, when using 7 keys with loss (theft) probability of 0.03 compared to 3 keys with loss (theft) probability of 0.01.



(a) $P^{theft} = 0.01$. **(b)** $P^{loss} = 0.01$.

Figure 7 Failure probability with different fault probabilities.

Table 3 Heterogeneous Keys.

	Key 1			Key 2			Key 3			$p_{failure}$	Wallet
	P_1^{loss}	P_1^{leak}	P_1^{theft}	P_2^{loss}	P_2^{leak}	P_2^{theft}	P_3^{loss}	P_3^{leak}	P_3^{theft}		
1	0.100	0.000	0.000	0.100	0.000	0.000	0.100	0.000	0.000	0.0010	$(1) \vee (2) \vee (3)$
2	0.000	0.100	0.000	0.000	0.100	0.000	0.000	0.100	0.000	0.0010	$(1 \wedge 2 \wedge 3)$
3	0.100	0.000	0.000	0.100	0.000	0.000	0.000	0.100	0.000	0.0100	$(1) \vee (2)$
4	0.100	0.000	0.000	0.000	0.100	0.000	0.000	0.100	0.000	0.0100	$(2 \wedge 3)$
5	0.050	0.050	0.000	0.050	0.050	0.000	0.050	0.050	0.000	0.0145	$(1 \wedge 2) \vee (1 \wedge 3) \vee (2 \wedge 3)$
6	0.050	0.050	0.000	0.050	0.050	0.000	0.000	0.100	0.000	0.0122	$(1 \wedge 3) \vee (2 \wedge 3)$
7	0.050	0.050	0.000	0.050	0.050	0.000	0.100	0.000	0.000	0.0122	$(1 \wedge 2) \vee (3)$
8	0.100	0.000	0.000	0.100	0.000	0.000	0.000	0.000	0.100	0.0100	$(1) \vee (2)$
9	0.000	0.000	0.100	0.000	0.000	0.100	0.000	0.000	0.100	0.0280	$(1 \wedge 2) \vee (1 \wedge 3) \vee (2 \wedge 3)$
10	0.000	0.000	0.050	0.000	0.000	0.050	0.000	0.000	0.050	0.0073	$(1 \wedge 2) \vee (1 \wedge 3) \vee (2 \wedge 3)$

5.3 Key Type Choice

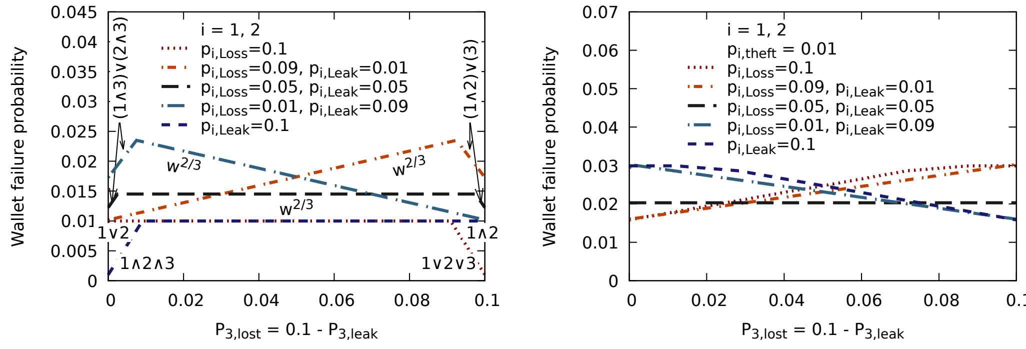
Apart from the number of keys, the owner could choose the types of keys she maintains. For example, she could keep 3 secrets in different safety deposit boxes, or three secrets on hardware devices held by different people. Alternatively, she could keep one key in a safety deposit box, another on a hardware device held by a person, and another memorized.

To compare such alternatives, assume the owner has a budget of 0.1 fault probability per key and can choose three keys such that for each key i: $P_i^{loss} + P_i^{leak} + P_i^{theft} = 0.1$. Table 3 shows the optimal wallet and its failure probability for different heterogeneous key sets. Among these settings, the best wallets are obtained with homogeneous keys that can either be only lost (Line 1) or leaked (Line 2), and theft is naturally the most problematic.

Figure 8 explores the effects of budget distribution with heterogeneous keys. For all key combinations, the sum of fault probabilities is 0.1. In all settings we keep the fault probabilities of keys 1 and 2 constant and vary that of key 3 such that its loss probability grows from 0 to 0.1 and its leak probability drops from 0.1 to 0.

In all cases, if the loss probability of k_3 is very low (and its leak probability is high) it is required in all combinations, as it is unlikely the owner would not have access to it (this is not always visible at this resolution). Similarly, if its leak probability is very low then the optimal wallet can be satisfied only with k_3.

With no theft (Figure 8a), if the probabilities for the first two keys are pure, i.e., only loss or only leak, it is best to have a third key of the same type. However, if they are not pure ($P^{loss} = 0.09, P^{leak} = 0.01$), it is better to have the third key of a different type (e.g., $P^{loss} = 0.01, P^{leak} = 0.09$).



(a) No theft, fault budget of 0.1.

(b) Theft in k_1 and k_2; fault budget of 0.1 for k_3.

Figure 8 Heterogeneous keys.

The optimal wallets change significantly if we introduce just a bit of theft probability (Figure 8b, $P_1^{theft} = P_2^{theft} = 0.01$), slightly increasing the fault budget. There is a significant advantage in taking k_3 to be different than the first two, i.e., if they have large loss probability, it is better to take k_3 with small loss probability. With the two keys having $P^{loss} = P^{leak} = 0.05$, all choices of k_3 result in the same wallet failure probability.

6 Conclusion

We present a simple model allowing, for the first time, to analyze the design of cryptocurrency wallets. Our analysis shows that careful design is necessary for constructing secure wallets – adding keys provides in general an exponential improvement, but strongly depends on parity; and choosing whether to add keys of the same type or of a different type depends on the exact key fault probabilities.

Our results raise questions for future work. First, user studies to quantify key fault probabilities are critically missing, both for theoretical analysis and to inform users; the infamous frequency of wallet failures indicates users commonly underestimate the fault probability of their keys. Secondly, the model can be expanded to consider (the undesirable) correlation between key faults. Those occur, for example, if a user keeps two keys at the same location and loses access to it, or if two company employees defect together. Thirdly, our evolutionary algorithm only reaches a limited number of keys, as the key state space became too large to evaluate. Techniques for estimating security with larger spaces would allow to analyze, or at least bound, wallet security with larger numbers of keys.

Nevertheless, the results presented here are of immediate importance to users, who can estimate their keys' fault probabilities, choose keys wisely, and thus improve the security of their funds.

References

1 Myrto Arapinis, Andriana Gkaniatsou, Dimitris Karakostas, and Aggelos Kiayias. A formal treatment of hardware wallets. In *International Conference on Financial Cryptography and Data Security*, pages 426–445. Springer, 2019.

2 Ali Bagherzandi, Stanislaw Jarecki, Nitesh Saxena, and Yanbin Lu. Password-protected secret sharing. In *Proceedings of the 18th ACM conference on Computer and Communications Security*, pages 433–444, 2011.

3 Praveen Baratam. Secure cryptocurrency depository. `https://www.coinvault.tech/wp-content/uploads/2020/10/CoinVault-Secure-Cryptocurrency-Depository.pdf`.

4 Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to better, how to make Bitcoin a better currency. In *Financial Cryptography and Data Security*, pages 399–414. Springer, Bonaire, 2012.

5 Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Research perspectives on Bitcoin and second-generation cryptocurrencies. In *Symposium on Security and Privacy*, San Jose, CA, USA, 2015. IEEE.

6 Joachim Breitner and Nadia Heninger. Biased nonce sense: Lattice attacks against weak ecdsa signatures in cryptocurrencies. In *International Conference on Financial Cryptography and Data Security*, pages 3–20. Springer, 2019.

7 Randolph Church. Nunmerical analysis of certain free distributive structures. *Duke Mathematical Journal*, 6(3):732–734, 1940. doi:10.1215/S0012-7094-40-00655-X.

8 Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 910–927. IEEE, 2020.

9 Richard Dedekind. Über zerlegungen von zahlen durch ihre grössten gemeinsamen theiler. In *Fest-Schrift der Herzoglichen Technischen Hochschule Carolo-Wilhelmina*. Springer, 1897.

10 Agoston E Eiben, James E Smith, et al. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

11 Shayan Eskandari, David Barrera, Elizabeth Stobert, and Jeremy Clark. A first look at the usability of bitcoin key management. In *Workshop on Usable Security and Privacy (USEC)*. Internet Society, 2015.

12 Benjamin Fabian, Tatiana Ermakova, Jonas Krah, Ephan Lando, and Nima Ahrary. Adoption of security and privacy measures in bitcoin–stated and actual behavior. *Available at SSRN 3184130*, 2018.

13 Dinei Florencio and Cormac Herley. Is everything we know about password stealing wrong? *IEEE Security & Privacy*, 10(6):63–69, 2012.

14 Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ecdsa with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1179–1194, 2018.

15 Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security. In *International Conference on Applied Cryptography and Network Security*, pages 156–174. Springer, 2016.

16 Andriana Gkaniatsou, Myrto Arapinis, and Aggelos Kiayias. Low-level attacks in bitcoin wallets. In *International Conference on Information Security*, pages 233–253. Springer, 2017.

17 Stanislaw Jarecki, Aggelos Kiayias, Hugo Krawczyk, and Jiayu Xu. Highly-efficient and composable password-protected secret sharing (or: How to protect your bitcoin wallet online). In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016.

18 Uri Kirstein, Shelly Grossman, Michael Mirkin, James Wilcox, Ittay Eyal, and Mooly Sagiv. Phoenix: A formally verified regenerating vault. *arXiv preprint*, 2021. arXiv:2106.01240.

19 Yehuda Lindell and Ariel Nof. Fast secure multiparty ecdsa with practical distributed key generation and applications to cryptocurrency custody. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1837–1854, 2018.

20 Malte Möser, Ittay Eyal, and Emin Gün Sirer. Bitcoin covenants. In *International conference on financial cryptography and data security*, pages 126–141. Springer, 2016.

21 Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

22 Marie Vasek, Joseph Bonneau, Ryan Castellucci, Cameron Keith, and Tyler Moore. The bitcoin brain drain: Examining the use and abuse of bitcoin brain wallets. In *International Conference on Financial Cryptography and Data Security*, pages 609–618. Springer, 2016.

23 Morgan Ward. Note on the order of the free distributive lattice, abstract 135. *Bull. Amer. Math. Soc*, 52, 1946.

24 Doug Wiedemann. A computation of the eighth dedekind number. *Order*, 8(1):5–6, 1991.

A Optimal Keys

For two cases we can easily prove what the optimal wallets are.

We often omit the function parameters to reduce clutter. For example, a wallet w and its sub-expressions are all functions of key availability, but rather than writing $w(k_1, k_2, k_3) = e_1(k_1, k_2, k_3) \lor e_2(k_1, k_2, k_3)$ we suffice with $w = e_1 \lor e_2$.

First, if keys can only suffer loss, the optimal wallet is the w_n^{OR} wallet.

▶ **Proposition 1.** *In a setting where all n keys can only suffer loss, i.e., $\forall 1 \leq i \leq n : P_i^{leak} = P_i^{theft} = 0$, for all wallets w, $p_{success}^P(w) \leq p_{success}^P(w_n^{\mathsf{OR}})$.*

Proof. Consider a wallet expressed as a DNF comprising m expressions, i.e., $w = e_1 \lor \ldots \lor e_m$, where each e_j is a conjunction of one or more key availabilities.

For any key k_i, consider the wallet w' formed by adding another disjunction of that key, i.e., $w'(k_1, \ldots, k_n) = w(k_1, \ldots, k_n) \lor k_i$.

Set a scenario σ with a positive probability $\Pr[\sigma] > 0$. Key i is either safe or lost. In both cases, it is not available to the adversary, therefore the adversary can satisfy w' iff it can satisfy w. If the owner can satisfy w then it can also satisfy w'. Therefore, if wallet w is successful in σ then wallet w' is also successful in σ. Therefore, the probability that the wallet w' is successful is not smaller than the probability that the wallet w is successful.

We take the wallet w and create a wallet $w_1(k_1, \ldots, k_n) = w(k_1, \ldots k_n) \lor k_1$, and continue creating a series of wallets such that $w_j(k_1, \ldots k_n) = w_{j-1}(k_1, \ldots k_n) \lor k_j$. As we have shown, each wallet is at least as successful as its predecessor and therefore at least as successful as w. The last wallet, $w_n(k_1, \ldots k_n) = w \lor k_1 \lor \cdots \lor k_n$, is equivalent to w_n^{OR}, completing our proof. ◀

Secondly, if keys can only suffer leakage, the optimal wallet is the w_n^{AND} wallet.

▶ **Proposition 2.** *In a setting where all n keys can only suffer leakage, i.e., $\forall 1 \leq i \leq n$: $P_i^{loss} = P_i^{theft} = 0$, for all wallets w, $p_{success}^P(w) \leq p_{success}^P(w_n^{\mathsf{AND}})$.*

The proof approach is similar.

Proof. Consider a wallet expressed as a DNF comprising m expressions, i.e., $w = e_1 \lor \ldots \lor e_m$, where each e_j is a conjunction of one or more key availabilities.

For any key k_i, consider the wallet w' formed by adding k_i to each conjunction, i.e., $w'(k_1, \ldots, k_n) = (e_1 \land k_i) \lor \cdots \lor (e_m \land k_i)$.

Set a scenario σ with a positive probability $\Pr[\sigma] > 0$. Key i is either safe or leaked. In both cases, the owner can access it, therefore the owner can satisfy w' iff it can satisfy w. If the adversary cannot satisfy w then it cannot satisfy w' either. Therefore, if wallet w is successful in σ then wallet w' is also successful in σ. Therefore, the probability that the wallet w' is successful is not smaller than the probability that the wallet w is successful.

We take the wallet $w = e_1 \lor \ldots e_m$ and create a wallet $w_1 = (e_1 \land k_1) \lor \cdots \lor (e_m \land k_1)$. and continue creating a series of wallets $w_1, \ldots, w_n$, where $w_j = e_1^j \lor \cdots \lor e_m^j$ and for all $2 \leq j \leq n$ and $1 \leq \ell \leq m$, we take $e_\ell^j = e_\ell^{j-1} \land k_j$.

As we have shown, each wallet is at least as successful as its predecessor and therefore at least as successful as w. The last wallet, $w_n = (e_1 \land k_1 \land \cdots \land k_n) \lor \cdots \lor (e_m \land k_1 \land \cdots \land k_n)$ is equivalent to w_n^{AND}, completing our proof. ◀

Secure Computation with Non-Equivalent Penalties in Constant Rounds

Takeshi Nakai
The University of Electro-Communications, Tokyo, Japan

Kazumasa Shinagawa
Ibaraki University, Ibaraki, Japan
National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Abstract

It is known that Bitcoin enables to achieve fairness in secure computation by imposing a monetary penalty on adversarial parties. This functionality is called *secure computation with penalties*. Bentov and Kumaresan (Crypto 2014) showed that it could be realized with $O(n)$ rounds and $O(n)$ broadcasts for any function, where n is the number of parties. Kumaresan and Bentov (CCS 2014) posed an open question: "Is it possible to design secure computation with penalties that needs only $O(1)$ rounds and $O(n)$ broadcasts?" In this work, we introduce *secure computation with non-equivalent penalties*, and design a protocol achieving this functionality with $O(1)$ rounds and $O(n)$ broadcasts only. The new functionality is the same as secure computation with penalties except that every honest party receives more than a predetermined amount of compensation while the previous one requires that every honest party receives the same amount of compensation. In particular, both are the same if all parties behave honestly. Thus, our result gives a partial answer to the open problem with a slight and natural modification of functionality.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases Secure computation, Fairness, Bitcoin

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2021.5

Funding This work was partially supported by JSPS KAKENHI Grant Numbers JP20J21248 and JP21K17702.

1 Introduction

1.1 Backgrounds

Secure computation enables parties to compute a function whose inputs are their private data [20]. There are several notions of security, such as privacy, correctness, independence of inputs, guaranteed output delivery, and fairness. Fairness requires that at the end of a protocol, either all parties learn the output value or none of them learn it, i.e., no malicious parties receive their output while some honest parties do not receive output. Unfortunately, it is known that secure computation cannot achieve fairness in the standard model if a majority of parties are corrupted [7].

In order to circumvent the impossibility result, there are works to achieve fairness in secure computation by imposing a monetary penalty on aborting parties [17]. We focus on achieving fairness by using decentralized digital currency [3, 2, 1, 4, 13], e.g., Bitcoin [18]. In secure computation on Bitcoin, an aborting party is given a penalty for losing coins, and honest parties are compensated with coins.

Back and Bentov [3] and Andrychowicz, Dziembowski, Malinowski, and Mazurek [2] introduced secure computation on Bitcoin. They studied fair lottery protocols that guarantee any party aborting after learning the result is forced to pay penalties to all other parties. After that, Bentov and Kumaresan [4] formalized such a model of computation as *secure* *computation with penalties.* In particular, they defined a *claim-or-refund functionality* $\mathcal{F}^*_{\mathrm{CR}}$ that plays an important role in secure computation with penalties. In $\mathcal{F}^*_{\mathrm{CR}}$, a sender can send coins with a puzzle $\phi_{s,r}(\cdot)$ and a round number τ to a receiver. The receiver gets the coins if he/she reveals a solution w such as $\phi_{s,r}(w) = 1$ in τ, and the sender gets back the coins if the receiver does not publish the solution in τ.

Bentov and Kumaresan [4] showed that secure computation with penalties can be realized for any function in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{CR}})$-hybrid model, where $\mathcal{F}_{\mathrm{OT}}$ is an ideal functionality of oblivious transfer. Their protocol requires $O(n)$ rounds[1] and $O(n)$ broadcasts, where n is the number of parties and the number of broadcasts is the number of transactions. Kumaresan and Bentov [13] introduced a new functionality $\mathcal{F}^*_{\mathrm{ML}}$ and showed that secure computation with penalties could be realized in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{ML}})$-hybrid model with only $O(1)$ rounds. However, this protocol requires $O(n^2)$ broadcasts. In the paper, they posed an open problem as follows:

Is it possible to design a fair protocol
that needs only $O(1)$ rounds and $O(n)$ broadcasts?

Related works

Kumaresan, Moran, and Bentov [15] extended secure computation with penalties to the *reactive model* that can also handle multistage functionalities, such as Texas Holdem poker. Also, they defined a new security model for the reactive model and proposed a fair protocol in the reactive model. This paper focuses on the single-stage (i.e., non-reactive) model following the same setting in [4].

Kumaresan and Bentov [14] improved the efficiency of protocols by amortizing the cost over multiple executions. Kumaresan, Vaikuntanathan, and Vasudevan [16] reduced the script complexity of $\phi_{s,r}$. This paper focuses on reducing the number of rounds and the number of broadcasts.

There are several works [11, 6, 8] based on the model with stateful contracts, which is stronger than our model. The model with stateful contracts can be instantiated by an advanced blockchain technique like Ethereum [19], while our model can be instantiated by Bitcoin.

1.2 Our Contributions

We introduce a new functionality, *secure computation with non-equivalent penalties*, which is a slightly relaxed variant of secure computation with penalties. It guarantees that each honest party is compensated with *more than* a predetermined amount of coins, while secure computation with penalties guarantees that every honest party is compensated for the *same* amount of coins. That is, in secure computation with non-equivalent penalties, two honest parties may be compensated with different amounts of coins, although they are at least a predetermined amount.

We show that secure computation with non-equivalent penalties can be realized for arbitrary functions in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{CR}})$-hybrid model (See Table 1). Our technical contribution is to propose a new *fair reconstruction protocol*, which is a subprotocol of a secure computation

[1] In [5], it is stated that one round should be about an hour on Bitcoin to prevent the double-spending attack. Thus, it implies that a s-round protocol takes about s hours.

Table 1 Comparison of secure computation protocols with (non-equivalent) penalties.

References	# of Rounds	# of Broadcasts	Compensation Amount
Bentov–Kumaresan [4]	$O(n)$	$O(n)$	Equivalent
Kumaresan-Bentov [13]	$O(1)$	$O(n^2)$	Equivalent
This work (Sect. 4)	$O(1)$	$O(n)$	Non-equivalent

protocol with (non-equivalent) penalties, with $O(1)$ rounds and $O(n)$ broadcasts. As a result, we obtain secure computation with non-equivalent penalties with $O(1)$ rounds and $O(n)$ broadcasts by replacing Bentov–Kumaresan's fair reconstruction protocol with ours.

We note that our protocol is equivalent to a protocol achieving secure computation with penalties if all parties behave honestly. Moreover, when we set the least amount of compensation appropriately, malicious behavior is prevented. We believe that our result gives a partial answer to the open problem posed by Kumaresan and Bentov [13].

2 Preliminaries

2.1 Basic Notations

For any positive integer $i \in \mathbb{N}$, $[i]$ denotes the set of integers $\{1, \ldots, i\}$. We denote by n the number of parties in a protocol. We denote by $H \subseteq [n]$ (resp. $C \subseteq [n]$) the set of honest (resp. corrupted) parties. Since each party is either honest or corrupted, it must hold $h + c = n$ for $h := |H|$ and $c := |C|$. We denote by k a security parameter. We assume that all parties are non-uniform probabilistic polynomial-time algorithms in k.

2.2 Secure Computation with Coins

Bentov–Kumaresan [4] introduced a new secure computation model called *secure computation with coins* (SCC) model. It is the same model as the standard model except that entities (i.e., parties, adversaries, ideal functionalities, and an environment) can deal with a non-standard entity called *coins*, which is an atomic entity representing electronic money. Coins are assumed to be having the following properties.

- Coins cannot be duplicated and forged.
- No multiple parties hold the same coin simultaneously.
- Any parties can transfer their coins to other parties freely.
- Each coin is perfectly indistinguishable from one another.

We use the notation $\mathsf{coins}(\cdot)$ to express the amount of coins. If a party owning $\mathsf{coins}(x)$ receives $\mathsf{coins}(y)$ from another party, then the party holds $\mathsf{coins}(x + y)$ as a result.

In the SCC model, some ideal functionalities can deal with coins. We call such a functionality a *special ideal functionality*. These functionalities are described with the superscript $*$, e.g., $\mathcal{F}^*_{\text{xxx}}$. We call an ideal functionality without handling coins a *standard* ideal functionality. Our protocol is realized in the hybrid model where parties have access to a standard functionality $\mathcal{F}_{\text{OT}}$, which is the ideal functionality for oblivious transfer, and a special ideal functionality $\mathcal{F}^*_{\text{CR}}$, described later.

The SCC model follows the real/ideal simulation paradigm as with the standard secure computation model. Let $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k, z)$ denote the output of an environment $\mathcal{Z}$ in the ideal world for realizing an ideal functionality $\mathcal{F}$, where $\mathcal{Z}$ (with an auxiliary input z) is interacting with an ideal adversary $\mathcal{S}$ on security parameter k. Let $\mathrm{HYBRID}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)$ denote the output of environment $\mathcal{Z}$ in the real (hybrid) world for executing a hybrid protocol π with an ideal functionality $\mathcal{G}$, where $\mathcal{Z}$ is interacting with a real adversary $\mathcal{A}$. The difference with the standard secure computation is that all entities (i.e., parties, adversaries, special ideal functionalities, and an environment) can deal with coins: sending coins, storing coins, and receiving coins.

▶ **Definition 1.** *Let π be a probabilistic polynomial-time n-party protocol and let $\mathcal{F}$ be a probabilistic polynomial-time n-party (standard or special) ideal functionality. We say that π* SCC *realizes $\mathcal{F}$ with abort in the $\mathcal{G}$ hybrid model (where $\mathcal{G}$ is a standard or special ideal functionality) if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ such that for every non-uniform probabilistic polynomial-time environment $\mathcal{Z}$, two families of random variables $\{\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^*}$ and $\{\mathrm{HYBRID}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^*}$ are computationally indistinguishable.*

2.3 Special Ideal Functionalities

2.3.1 Claim-or-refund functionality $\mathcal{F}^*_{\mathrm{CR}}$

This functionality $\mathcal{F}^*_{\mathrm{CR}}$ [4] can be seen as an analogue of puzzles with bounty. Roughly speaking, for a puzzle $\phi_{s,r}$ with coins submitted by a sender, a receiver gets the coins if and only if he/she submits a solution w of the puzzle (i.e., $\phi_{s,r}(w)=1$). $\mathcal{F}^*_{\mathrm{CR}}$ consists of three phases: deposit, claim, and refund. In the deposit phase, a sender P_s sends to a receiver P_r "conditional" coins together with a circuit $\phi_{s,r}$. The coins also have a round number τ specified by the sender. In the claim phase, the receiver P_r claims to receive the coins. P_r can receive the coins only if he/she broadcasts the witness w of $\phi_{s,r}$ (i.e., $\phi_{s,r}(w)=1$) in τ. Note that the witness w published in the claim phase is made public to all parties. In the refund phase, if P_r does not claim in τ, then the coins are refunded to the sender P_s. See Algorithm 1 for a formal description of $\mathcal{F}^*_{\mathrm{CR}}$.[2] At least one broadcast is necessary to realize $\mathcal{F}^*_{\mathrm{CR}}$ on Bitcoin. Thus, the number of calling $\mathcal{F}^*_{\mathrm{CR}}$ corresponds to the number of broadcasts.

We call the message in the deposit phase a deposit transaction. We use the following "arrow" notation to denote the deposit transaction for the sender P_s and the receiver P_r.

$$P_s \xrightarrow[c,\tau]{w} P_r$$

After making an arrow from P_s to P_r as above (i.e., after the deposit phase), P_r can claim to receive $\mathsf{coins}(c)$ only if he/she publishes the witness w in round τ. $\mathsf{coins}(c)$ is refunded back to the original holder P_s if P_r does not publish w in τ.

2.3.2 Secure computation with penalties $\mathcal{F}^*_f$

This functionality $\mathcal{F}^*_f$ is the same as the standard secure function evaluation except that aborting parties are forced to pay penalties [4]. In principle, it guarantees the following properties:

- no honest party pays any penalty, and
- if a party aborts after learning the output value and does not tell the value to the other parties, then every party who does not learn the value is compensated with coins.

[2] [5, 14] show how to realize $\mathcal{F}^*_{\mathrm{CR}}$ using Bitcoin.

Algorithm 1 Claim-or-refund functionality $\mathcal{F}^*_{\mathrm{CR}}$ [4].

```
Setup The session identifier is $sid$. Running with parties $P_1, \dots, P_n$ and an ideal adversary $\mathcal{S}$.
Deposit phase Receiving $(\text{deposit}, sid, ssid, s, r, \phi_{s,r}, \tau, \textsf{coins}(c))$ from $P_s$, perform the following process.
    1) Record the message $(\text{deposit}, sid, ssid, s, r, \phi_{s,r}, \tau, c)$
    2) Send all parties $(\text{deposit}, sid, ssid, s, r, \phi_{s,r}, \tau, c)$
        - Ignore any future messages with the same $ssid$ from $P_s$ to $P_r$.
Claim phase Receiving $(\text{claim}, sid, ssid, s, r, \phi_{s,r}, \tau, c, w)$ from $P_r$ in round $\tau$, perform the following process.
    1) Check the two conditions:
        - $(\text{deposit}, sid, ssid, s, r, \phi_{s,r}, \tau, c)$ was recorded,
        - $\phi_{s,r}(w) = 1$.
    2) If both checks are passed, perform the following process:
        2-i) send $(\text{claim}, sid, ssid, s, r, \phi_{s,r}, \tau, c, w)$ to all parties,
        2-ii) send $(\text{claim}, sid, ssid, s, r, \phi_{s,r}, \tau, \textsf{coins}(c))$ to $P_r$,
        2-iii) delete the record $(\text{deposit}, sid, ssid, s, r, \phi_{s,r}, \tau, c)$.
Refund phase In $\tau + 1$, if the record $(\text{deposit}, sid, ssid, s, r, \phi_{s,r}, \tau, c)$ was not deleted, then perform the following process:
    1) send $(\text{refund}, sid, ssid, s, r, \phi_{s,r}, \tau, \textsf{coins}(c))$ to $P_s$,
    2) delete the record $(\text{deposit}, sid, ssid, s, r, \phi_{s,r}, \tau, c)$.
```

See Algorithm 2 for a formal description of $\mathcal{F}^*_f$. The parameters q and d specify the amounts of coins. At the beginning of the protocol, each party submits $\textsf{coins}(d)$ together with input x_i. If a party aborts after learning the output and does not tell the value to the other parties, then $\mathcal{F}^*_f$ gives $\textsf{coins}(q)$ to every party who does not learn the output as compensation. Then, it is important note that the compensation amount is always q for any parties.

H is a set of honest parties and $H' \subseteq H$ is a subset chosen by $\mathcal{S}$, which represents parties who are compensated. At first glance, it is somewhat strange that $\mathcal{S}$ chooses a subset of honest parties. The reason why H' is needed is that there are two types of aborting in secure computation with abort. The first one is that an adversary aborts after obtaining the output and thus honest parties cannot obtain the outputs. In this case, $\mathcal{S}$ chooses $H' = H$ and all honest parties are compensated with $\textsf{coins}(q)$ although the output is stolen by the adversary. The second one is that an adversary aborts before obtaining the output so the protocol just terminates. In this case, $\mathcal{S}$ chooses $H' \subsetneq H$ (possibly empty) and the parties in H' are compensated with $\textsf{coins}(q)$.[3]

2.4 Non-malleable secret sharing with public verifiability and public reconstructibility

A *non-malleable secret sharing scheme with public verifiability and public reconstructibility* (in short, pubNMSS) [4] is a variant of non-malleable secret sharing scheme. The share algorithm of pubNMSS takes a secret s as input, generates "tag-token" pairs $(\textsf{Tag}_i, \textsf{Token}_i)_{i\in[n]}$, and

[3] H'' is required for a technical reason in order to prove the security. See [4] for a detail. In order to prove the security of our protocol, our new functionality follows the same strategy.

Algorithm 2 Secure computation with penalties $\mathcal{F}^*_f$ [4].

```
Setup The session identifier is $sid$. Running with parties $P_1, \ldots, P_n$, and an ideal adversary
    $\mathcal{S}$ that corrupts parties $\{P_i\}_{i \in C}$. Let $d$ be a parameter representing the safety deposit,
    and let $q$ denote the penalty amount.
Input phase Wait to receive the following messages.
    - $(\text{input}, sid, ssid, i, x_i, \mathsf{coins}(d))$ from $P_i$ for all $i \in H$
    - $(\text{input}, sid, ssid, \{y_i\}_{i \in C}, H', \mathsf{coins}(h'q))$ from $\mathcal{S}$, where $H' \subseteq H$ and $h' = |H'|$
Output phase Perform the following process.
    1) Send $(\text{return}, sid, ssid, \mathsf{coins}(d))$ to each $P_i$ for $i \in H$.
    2) Compute $(y_1, \ldots, y_n) \leftarrow f(x_1, \ldots, x_n)$.
        - if $h' = 0$, then send message $(\text{output}, sid, ssid, y_i)$ to $P_i$ for $i \in H$, and terminate.
        - If $0 < h' < h$, then send $(\text{extra}, sid, ssid, \mathsf{coins}(q))$ to $P_i$ for each $i \in H'$, and
          terminate, where $h := |H|$.
        - If $h' = h$, then send message $(\text{output}, sid, ssid, \{y_i\}_{i \in C})$ to $\mathcal{S}$.
    3) If $\mathcal{S}$ returns $(\text{continue}, sid, ssid, H'')$, where $H'' \subseteq H$, then perform the following
       process:
         3-i) send $(\text{output}, sid, ssid, y_i)$ to $P_i$ for all $i \in H$,
        3-ii) send $(\text{payback}, sid, ssid, \mathsf{coins}((h - h'')q))$ to $\mathcal{S}$ where $h'' = |H''|$,
       3-iii) send $(\text{extrapay}, sid, ssid, \mathsf{coins}(q))$ to $P_i$ for each $i \in H''$.
    4) Else if $\mathcal{S}$ returns $(\text{abort}, sid, ssid)$, send $(\text{penalty}, sid, ssid, \mathsf{coins}(q))$ to $P_i$ for all $i \in H$.
```

outputs Token_i and $(\mathsf{Tag}_1, \ldots, \mathsf{Tag}_n)$ to each party P_i. The parties can reconstruct s by collecting all n tokens. For all $i \in [n]$, the parties can verify if the published Token_i is valid with Tag_i. The tag-token pairs have the following properties:

- all tags $(\mathsf{Tag}_1, \ldots, \mathsf{Tag}_n)$ leak no information about s,
- any sets of $t(< n)$ tokens leak no information about s,
- for any $i \in [n]$, the adversary cannot generate $\mathsf{Token}'_i (\neq \mathsf{Token}_i)$ such that $(\mathsf{Tag}_i, \mathsf{Token}'_i)$ is a valid tag-token pair.

A pubNMSS scheme can be obtained from the *honest-binding commitment*, which can be constructed from one-way functions [9]. Tag_i is an (honest-binding) commitment that is computed by a secret share sh_i and a randomness r_i as input, and $\mathsf{Token}_i := (sh_i, r_i)$. Namely, the parties can verify if the published $\mathsf{Token}'_i = (sh'_i, r'_i)$ is valid by comparing Tag_i and the commitment whose input is sh'_i and r'_i. In the following discussions, this verification corresponds to $\phi_{s,r}$ in $\mathcal{F}^*_{\mathrm{CR}}$ executions.

3 Existing Protocol for secure computation with penalties

In this section, we introduce Bentov–Kumaresan's protocol [4] for secure computation with penalties in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{CR}})$-hybrid model.

3.1 Bentov–Kumaresan's Protocol

For a function f, an *augmented function* denoted by $\hat{f}$ is defined by a function that takes an input x and distributes secret shares of the output value $f(x)$. The underlying secret sharing scheme is non-malleable secret sharing with publicly verifiability and publicly reconstructibility (Section 2.4). Thus the augmented function $\hat{f}$ outputs a token Token_i (i.e., a share of $f(x)$) and a set of tags $(\mathsf{Tag}_1, \ldots, \mathsf{Tag}_n)$ to party P_i.

Bentov–Kumaresan's protocol proceeds as follows:

(i) The parties execute a secure computation protocol for $\hat{f}$, and then each party P_i obtains a token Token_i of $f(x)$ and a set of tags $(\mathsf{Tag}_1, \dots, \mathsf{Tag}_n)$. (Note that this is the standard computation without Bitcoin).

(ii) For the reconstruction of tokens, the parties execute the *fair reconstruction protocol*, where each party P_i is forced to broadcast a token Token_i. The validity of the submitted token Token_i is verified with the tag Tag_i. (Note that this computation is based on Bitcoin).

It is well known that the OT functionality $\mathcal{F}_{\mathrm{OT}}$ is sufficient to achieve secure computation for any standard functionality [12, 10]. Moreover, this can be performed in constant rounds [10]. Therefore, the secure computation stage (i) is performed in constant rounds in the $\mathcal{F}_{\mathrm{OT}}$-hybrid model.

The main step of Bentov–Kumaresan's protocol is the fair reconstruction protocol (ii). By collecting all tokens, the parties can reconstruct the output value $f(x)$. However, malicious parties may abort so as to learn the output value while other parties do not. The fair reconstruction protocol prevents parties from aborting in the reconstruction phase. When malicious parties abort, they have to pay some amount of money for compensation to honest parties. It satisfies the following conditions:

(A) No honest party pays any penalty.

(B) If an adversary learns the reconstruction result, but an honest party cannot, then the honest party is compensated with coins. Furthermore, the compensation amounts are the same for any honest parties.

Note that honest parties are not guaranteed to receive compensation if an adversary aborts without learning the output value.

In summary, secure computation with penalties can be realized by executing a secure computation protocol for $\hat{f}$ and the fair reconstruction protocol. The next section shows Bentov–Kumaresan's fair reconstruction protocol in the $\mathcal{F}^*_{\mathrm{CR}}$-hybrid model.

3.2 Bentov–Kumaresan's Fair Reconstruction Protocol

Hereafter, we use T_i to denote Token_i. Suppose that each party P_i has a token T_i and a set of tags $(\mathsf{Tag}_1, \dots, \mathsf{Tag}_n)$ at the beginning of the fair reconstruction protocol. We assume that all parties agree on the penalty amount q, where honest parties are compensated with $\mathsf{coins}(q)$ when malicious parties abort with obtaining the output value. In the below, we successively explain a naïve approach, a solution for the two-party setting, and a solution for the n-party setting.

Naïve approach. Suppose that the number of parties is two. A naïve approach is to make a deposit transaction from P_1 to P_2 and a deposit transaction of the reverse direction as follows:

$$P_1 \xrightarrow[q,\tau]{T_2} P_2 \tag{1}$$

$$P_2 \xrightarrow[q,\tau]{T_1} P_1 \tag{2}$$

The above arrow means that "P_2 can receive $\mathsf{coins}(q)$ only if P_2 publishes the token T_2, otherwise $\mathsf{coins}(q)$ is refunded back to P_1" (see Section 2.3). The bottom arrow is similar. At first glance, it seems a fair reconstruction protocol satisfying conditions (A) and (B) in Section 3.1. However, it is not the case. For instance, when P_2 is malicious, P_2 can steal $\mathsf{coins}(q)$ from P_1 as follows: after establishing transaction (1), P_2 publishes the token T_2 without making transaction (2). As a result, honest P_1 loses $\mathsf{coins}(q)$. This violates condition (A).

Bentov-Kumaresan's solution. In order to avoid the above attack, Bentov–Kumaresan's fair reconstruction protocol for the two-party setting proceeds as follows:

$$P_1 \xrightarrow[q,\tau_2]{T_1 \wedge T_2} P_2 \tag{1}$$

$$P_2 \xrightarrow[q,\tau_1]{T_1} P_1 \tag{2}$$

where the rounds satisfy $\tau_1 < \tau_2$. (Hereafter, we assume $\tau_i < \tau_{i+1}$ for any integer i.) P_1 first makes a deposit transaction for $T_1 \wedge T_2$. Transaction (1) means that P_2 can receive $\mathsf{coins}(q)$ only if P_2 publishes both T_1 and T_2 in τ_2. Namely, it is necessary that both (Tag_1, T_1) and (Tag_2, T_2) are valid tag-token pairs to satisfy $\phi_{s,r}(T_1 \wedge T_2) = 1$. After making the first deposit transaction, P_2 makes a deposit transaction for T_1. Transaction (2) means that P_1 can receive $\mathsf{coins}(q)$ only if P_1 publishes T_1 in τ_1. In the claim phase, P_1 first publishes T_1, and then P_2 publishes both T_1 and T_2.

It is important to note that P_1 needs to make transaction (1) first. As a result, P_2 cannot claim this transaction without making transaction (2) since P_2 does not know T_1 yet. Also, the claims are performed in the reverse order of making the transactions, i.e., P_1 first claims.

If P_2 aborts after P_1 claims, then P_2 is penalized with $\mathsf{coins}(q)$ and P_1 is compensated with that coins. Thus, P_2 needs to publish T_2 in order not to lose $\mathsf{coins}(q)$. Also, both parties never are penalized if they behave honestly. Therefore, the above protocol satisfies the conditions (A) and (B) in Section 3.1.

We show Bentov-Kumaresan's solution for the n-party setting on the left side of Figure 1. As with the two-party case, parties make deposit transactions from the top and claim from the bottom in the n-party setting. Namely, the parties make transactions (1) to $(2n-2)$ and claim transactions $(2n-2)$ to (1).

Here, we describe an intuitive explanation that Bentov-Kumaresan's fair reconstruction protocol satisfies the condition (A) and (B). (See [5] for a formal security proof based on Definition 1.) It is trivial that no party loses coins if all parties behave honestly. Thus, we consider the case where there is a party to abort.

Let consider the case where an adversary aborts in the deposit phase. Since no honest party publishes his/her token, the adversary does not learn the reconstruction result nor receives any coins from honest parties. This case satisfies the condition (A) and (B).

Let consider the case where an adversary aborts in the claim phase. In order to learn the reconstruction result, the adversary must collude all parties that have not claimed yet to learn tokens that are not published. Every honest party holds $\mathsf{coins}(q)$ since he/she has already claimed and has got coins. This case also satisfies the condition (A) and (B).

Efficiency. Bentov–Kumaresan's fair reconstruction protocol requires n rounds for deposit phase and n rounds for claim phase, and thus it requires a total of $2n$ rounds. Also, it requires $2n-2$ calls of $\mathcal{F}^*_{\mathrm{CR}}$. Recall that the augmented function can be computed in a constant round for any function. Therefore, for any function, Bentov–Kumaresan's protocol for the secure computation with penalties can be SCC realized in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{CR}})$-hybrid model with $O(n)$ rounds and $O(n)$ calls of $\mathcal{F}^*_{\mathrm{CR}}$.

Table 2 Comparison of fair reconstruction protocols.

References	# of Rounds	# of Calling $\mathcal{F}^*_{\mathrm{CR}}$	Compensation Amount
Bentov–Kumaresan [4]	$2n$	$2n-2$	Equivalent
This work (Sect. 4)	8	$3n-4$	Non-equivalent

Algorithm 3 Secure computation with non-equivalent penalties $\mathcal{F}^*_{f,\mathrm{neq}}$.

Setup The session identifier is *sid*. Running with parties $P_1, \ldots, P_n$, and an ideal adversary $\mathcal{S}$ that corrupts parties $\{P_i\}_{i \in C}$. Let d be a parameter representing the safety deposit. Let q denote the minimum penalty amount.

Input phase Wait to receive the following messages.

- $(\mathrm{input}, sid, ssid, i, x_i, \mathsf{coins}(d))$ from P_i for all $i \in H$
- $(\mathrm{input}, sid, ssid, \{x_i\}_{i \in C}, H', \mathsf{coins}(\sum_{i \in H'} q_i))$ from $\mathcal{S}$, where $H' \subseteq H$ and $q_i\ (\geq q)$ is the penalty amount for each $i \in H'$.

Output phase Perform the following process.

```
1) Send $(\mathrm{return}, sid, ssid, \mathsf{coins}(d))$ to each $P_r$ for $r \in H$.
2) Compute $(y_1, \ldots, y_n) \leftarrow f(x_1, \ldots, x_n)$.
   - if $h' = 0$, then send message $(\mathrm{output}, sid, ssid, z_r)$ to $P_r$ for $r \in H$, and terminate.
   - If $0 < h' < h$, then send $(\mathrm{extra}, sid, ssid, \mathsf{coins}(q_i))$ to $P_i$ for each $i \in H'$, and terminate, where $h := |H|$.
   - If $h' = h$, then send message $(\mathrm{output}, sid, ssid, \{y_i\}_{i \in C})$ to $\mathcal{S}$.
3) If $\mathcal{S}$ returns $(\mathrm{continue}, sid, ssid, H'')$, where $H'' \subseteq H$, then perform the following process:
   3-i) send $(\mathrm{output}, sid, ssid, y_i)$ to $P_i$ for all $i \in H$,
   3-ii) send $(\mathrm{payback}, sid, ssid, \mathsf{coins}(\sum_{i \in H'} q_i - \sum_{j \in H''} q_j)$ to $\mathcal{S}$ where $h'' = |H''|$,
   3-iii) send $(\mathrm{extrapay}, sid, ssid, \mathsf{coins}(q_i))$ to $P_i$ for each $i \in H''$.
4) Else if $\mathcal{S}$ returns $(\mathrm{abort}, sid, ssid)$, send $(\mathrm{penalty}, sid, ssid, \mathsf{coins}(q_i))$ to $P_i$ for each $i \in H$.
```

4 Proposed Protocol

In this section, we introduce a special functionality called secure computation with non-equivalent penalties (Section 4.1). Then we design a protocol achieving this functionality in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{CR}})$-hybrid model. In particular, we design a new fair reconstruction protocol in the $\mathcal{F}^*_{\mathrm{CR}}$-hybrid model (Section 4.3), and putting it with a secure computation protocol for an augmented function into the $\mathcal{F}_{\mathrm{OT}}$-hybrid model as in Section 3.1. Notably, our protocol requires $O(1)$ rounds and $O(n)$ broadcasts only (See Table 1).

4.1 Secure Computation with Non-equivalent Penalties

In secure computation with penalties $\mathcal{F}^*_f$, all honest parties are compensated with the same amount of money $\mathsf{coins}(q)$. A new functionality, secure computation with non-equivalent penalties $\mathcal{F}^*_{f,\mathrm{neq}}$, is the same as $\mathcal{F}^*_f$ except that each honest party is compensated with $\mathsf{coins}(q)$ *or more*, i.e., the amount of compensation *may* be different with each party. For example, in $\mathcal{F}^*_{f,\mathrm{neq}}$, we allow the following situation: An honest P_1 is compensated with $\mathsf{coins}(q)$ but an honest P_2 is compensated with $\mathsf{coins}(2q)$.

See Algorithm 3 for a formal definition of $\mathcal{F}^*_{f,\mathrm{neq}}$. The difference with $\mathcal{F}^*_f$ is that a simulator can decide the amount q_i for each $i \in H'$ and inputs $\mathsf{coins}(\sum_{i\in H'} q_i)$ while a simulator in $\mathcal{F}^*_f$ must input $\mathsf{coins}(h'q)$ for $h' := |H'|$. We require that $q_i \geq q$ for all $i \in H'$, where q is the minimum amount of compensation.

We note that compensation happens only when a malicious party has stolen the output value. That is, $\mathcal{F}^*_f$ and $\mathcal{F}^*_{f,\mathrm{neq}}$ are the same if all parties behave honestly. By choosing q appropriately, it is possible to prevent malicious behavior, and then we obtain a protocol with fairness. In this sense, a new functionality $\mathcal{F}^*_{f,\mathrm{neq}}$ brings almost the same effect on $\mathcal{F}^*_f$.

4.2 Fair Reconstruction for Secure Computation with Non-equivalent Penalties

Following Bentov-Kumaresan's protocol, we construct a fair reconstruction protocol to realize secure computation with non-equivalent penalties. In order to realize secure computation with non-equivalent penalties, a fair reconstruction protocol needs to satisfy the following conditions:

(A) No honest party pays any penalty.

(B*) If an adversary learns the reconstruction result, but an honest party cannot, then the honest party is compensated with coins. Furthermore, the compensation is more than a predetermined amount.

Note that the difference between condition (B*) and condition (B) in Section 3.1 is the amount of compensations only. Namely, our fair reconstruction protocol does not guarantee that each honest party is compensated with the same amount of coins.

4.3 Our Fair Reconstruction Protocol

Our fair reconstruction protocol proceeds as follows (see also the right side of Figure 1):

Deposit phase

1) For $i \in \{1, \dots, n-1\}$, P_i makes a transaction to send P_n $\mathsf{coins}(q)$ with a circuit $\phi_{i,n}$ and a round number τ_4, where $\phi_{i,n}(x) = 1$ only if $x = T_1 \wedge \cdots \wedge T_n$.

2) P_n makes a transaction to send P_{n-1} $\mathsf{coins}((n-1)q)$ with a circuit $\phi_{n,n-1}$ and a round number τ_3, where $\phi_{n,n-1}(x) = 1$ only if $x = T_1 \wedge \cdots \wedge T_{n-1}$.

3) For $i \in \{1, \dots, n-2\}$, P_{n-1} makes a transaction to send P_i $\mathsf{coins}((n-1)q)$ with a circuit $\phi_{n-1,i}$ and a round number τ_2, where $\phi_{n-1,i}(x) = 1$ only if $x = T_{n-1} \wedge T_i$.

4) For $i \in \{1, \dots, n-2\}$, P_i makes a transaction to send P_{n-1} $\mathsf{coins}((n-2)q)$ with a circuit $\phi_{i,n-1}$ and a round number τ_1, where $\phi_{i,n-1}(x) = 1$ only if $x = T_{n-1}$.

Claim phase

5) P_{n-1} claims by publishing T_{n-1} in round τ_1 and receives $\mathsf{coins}((n-2)q)$ from each of $P_1, \dots, P_{n-2}$.

6) For $i \in \{1, \dots, n-2\}$, P_i claims by publishing $T_{n-1} \wedge T_i$ in round τ_2 and receives $\mathsf{coins}((n-1)q)$ from P_{n-1}.

7) P_{n-1} claims by publishing $T_1 \wedge \cdots \wedge T_{n-1}$ in round τ_3 and receives $\mathsf{coins}((n-1)q)$ from P_n.

8) P_n claims by publishing $T_1 \wedge \cdots \wedge T_n$ in round τ_4 and receives $\mathsf{coins}(q)$ from each of $P_1, \dots, P_{n-1}$.

Our fair reconstruction protocol requires eight rounds and $3n-4$ calls of $\mathcal{F}^*_{\mathrm{CR}}$. Since $\mathcal{F}_{\mathrm{OT}}$ is sufficient to compute any standard functionality in constant rounds, we can derive the following theorem. (We defer the proof to the full version.)

▶ **Theorem 2.** *Assuming the existing of one-way functions, for every n-party functionality f there exists a protocol that* SCC *realizes $\mathcal{F}^*_{f,\mathrm{neq}}$ in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{CR}})$-hybrid model. The protocol requires $O(1)$ rounds and $O(n)$ calls of $\mathcal{F}^*_{\mathrm{CR}}$.*

4.4 Idea behind Our Protocol

See Figure 2 that shows flows of the claim phase of Bentov–Kumaresan's fair reconstruction protocol and ours.[4] In Bentov–Kumaresan's protocol, parties publishes his/her token in serial order, i.e., each token is published in each round. (Token T_i is published in round τ_i.) Thus, their protocol requires $O(n)$ rounds.

On the other hand, our protocol enables to publish multiple tokens in one round to improve the round complexity. See step 6) in Section 4.3, the parties $P_1, \ldots, P_{n-2}$ publish their token in one round.

In the claim phase, our protocol proceeds as follows: We call $P_1, \ldots, P_{n-2}$ *middle parties* and P_{n-1} *aggregator.* In round τ_1, the aggregator P_{n-1} collects coins from all middle parties by publishing token T_{n-1}. After that, the middle parties publishes their tokens $T_1, \ldots, T_{n-2}$ and receive coins, which are more than they sent in round τ_1, from the aggregator P_{n-1} in round τ_2. In round τ_3, the aggregator P_{n-1} receives coins from P_n by publishing his/her token and all of middle parties' tokens. In the last round τ_4, P_n publishes the last token T_n and receives coins from every other party. As a result, all parties learn the reconstruction result and every party's wallet are balanced, i.e., it has neither loss nor gain.

We discuss the amount of coins sent in each transaction to satisfy the conditions (A) and (B*) below.

The amount of coins. In our protocol, P_n receives $\mathsf{coins}(q)$ from every other party in the last round τ_4. (See Figure 1.) In order to satisfy the condition (A), every wallet of $P_1, \ldots, P_{n-1}$ must hold $\mathsf{coins}(q)$ at the end of round τ_3. We show that our protocol satisfies this condition in Figure 3.

When we decide the amount of coins in rounds τ_1 and τ_2, we should note that the aggregator P_{n-1} cannot claim in round τ_3 if at least one of the middle parties abort in round τ_2. Since the aggregator sends more coins in round τ_2 than he/she received in round τ_1, his/her wallet holds negative amount of coins at the end of round τ_2. In order to satisfy the conditions (A) and (B*), it is necessary to satisfy that the aggregator's wallet holds positive amount of coins at the end of round τ_2 if at least one of the middle parties abort in round τ_2. The amounts of coins sent in rounds τ_1 and τ_2 are derived as follows.

Suppose that P_{n-1} gets $\mathsf{coins}(xq)$ from each of $P_1, \ldots, P_{n-2}$ in round τ_1, and each of $P_1, \ldots, P_{n-2}$ get $\mathsf{coins}((x+1)q)$ from P_{n-1} in round τ_2. In round τ_2, P_{n-1}'s wallet should have positive amount of coins unless all of $P_1, \ldots, P_{n-2}$ claims. Thus, we can derive x from the following equation: $(n-2)x > (n-3)(x+1)$. The least solution of the equation is $x = n-2$. Therefore, each middle party sends $\mathsf{coins}((n-2)q)$ to the aggregator in round τ_1 and the aggregator sends $\mathsf{coins}((n-1)q)$ to each middle party in round τ_2.

Security intuition. Let consider the case where one of the middle parties aborts in round τ_2. (See Figure 4.) Suppose that P_1 aborts in round τ_2, i.e., he/she does not publish T_1 and does not receive $\mathsf{coins}((n-1)q)$ from P_{n-1}. Note that P_1 must collude with P_n to learn the

[4] For ease of understanding, Figure 2 omits the transactions (among P_n and other parties) in the last round. (See transactions $(1), (2), \ldots, (n-1)$ in Figure 1.)

reconstruction result. Thus, the condition (B*) is satisfied since every wallet of $P_2, \dots, P_{n-1}$ holds $\mathsf{coins}(q)$ as the compensation at the end of the protocol. Furthermore, since no honest party does not pay a penalty, the condition (A) is satisfied. We can confirm that our protocol satisfies the conditions (A) and (B*) by the same way in the other cases.

▶ Remark 3. Compensations to honest parties may not be the same amount of coins. See P_{n-1} who receives $\mathsf{coins}((n-2)q)$ from each of $P_1, \dots, P_{n-2}$ in round τ_1. The amount of P_{n-1}'s compensation depends on the number of aborting parties in them. On the other hand, compensations for other parties are $\mathsf{coins}(q)$. Namely, P_{n-1} is the only party who can be compensated with more than $\mathsf{coins}(q)$.

▶ Remark 4. At first glance, it seems that rounds τ_3 and τ_4 need not be separated since P_n can already claim in τ_3. However, if these rounds are combined into one (i.e., $\tau_4 = \tau_3$), the modified protocol violates condition (A). Suppose all but P_n are malicious. First, in the deposit phase, the adversary makes the $n-1$ transactions to P_n honestly. However, after P_n makes the deposit transaction to P_{n-1}, the adversary waits for time to pass without making the subsequent transactions. Just before the end of τ_3, the adversary claims the transaction made by P_n and obtains $\mathsf{coins}((n-1)q)$. P_{n-1} can get that coins back by claiming $n-1$ transactions made by the adversary, however P_n may not claim due to the lack of time remaining. As a result, P_n may lose the coins, which violates the condition (A).

5 Conclusion

This paper focused on Bentov and Kumaresan's work [4] in secure computation with penalties. They showed that secure computation with penalties could be constructed with $O(n)$ rounds and $O(n)$ broadcasts for any function in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{CR}})$-hybrid model. Also, it was the open problem whether the round order could be improved to $O(1)$ with $O(n)$ broadcasts [13].

This paper presented a positive answer to this question in a relaxed setting. In Bentov-Kumaresan's protocol, every honest party can be compensated with the same amount of coins when an adversary aborts after learning the output value. On the other hand, in our setting, every honest party is guaranteed to be compensated with more than a predetermined amount of coins, but not the same amount. We formalized this new setting as secure computation with non-equivalent penalties. We showed that secure computation with non-equivalent penalties could be realized with $O(1)$ rounds and $O(n)$ broadcasts for arbitrary functions in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{CR}})$-hybrid model. In particular, we improved the fair reconstruction protocol [4], which is a key ingredient for realizing secure computation with penalties.

References

1 Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Łukasz Mazurek. Fair two-party computations via bitcoin deposits. In Rainer Böhme, Michael Brenner, Tyler Moore, and Matthew Smith, editors, *Financial Cryptography and Data Security*, pages 105–121, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

2 Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Secure multiparty computations on bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 443–458, 2014. `doi:10.1109/SP.2014.35`.

3 Adam Back and Iddo Bentov. Note on fair coin toss via bitcoin. *CoRR*, abs/1402.3698, 2014. `arXiv:1402.3698`.

4 Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, pages 421–439, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

5 Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. Cryptology ePrint Archive, Report 2014/129, 2014.

6 Iddo Bentov, Ranjit Kumaresan, and Andrew Miller. Instantaneous decentralized poker. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 410–440, Cham, 2017. Springer International Publishing.

7 R Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, pages 364–369, New York, NY, USA, 1986. Association for Computing Machinery. `doi:10.1145/12130.12168`.

8 Bernardo David, Rafael Dowsley, and Mario Larangeira. Kaleidoscope: An efficient poker protocol with payment distribution and penalty enforcement. In Sarah Meiklejohn and Kazue Sako, editors, *Financial Cryptography and Data Security*, pages 500–519, Berlin, Heidelberg, 2018. Springer Berlin Heidelberg.

9 Juan A. Garay, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. Adaptively secure broadcast, revisited. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC '11, pages 179–186, New York, NY, USA, 2011. Association for Computing Machinery. `doi:10.1145/1993806.1993832`.

10 Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer – efficiently. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, pages 572–591, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

11 Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Fair and robust multi-party computation using a global transaction ledger. In *Proceedings, Part II, of the 35th Annual International Conference on Advances in Cryptology – EUROCRYPT 2016 - Volume 9666*, pages 705–734, Berlin, Heidelberg, 2016. Springer-Verlag.

12 Joe Kilian. Founding crytpography on oblivious transfer. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 20–31, New York, NY, USA, 1988. Association for Computing Machinery. `doi:10.1145/62212.62215`.

13 Ranjit Kumaresan and Iddo Bentov. How to use bitcoin to incentivize correct computations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 30–41, New York, NY, USA, 2014. Association for Computing Machinery. `doi:10.1145/2660267.2660380`.

14 Ranjit Kumaresan and Iddo Bentov. Amortizing secure computation with penalties. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 418–429, New York, NY, USA, 2016. Association for Computing Machinery. `doi:10.1145/2976749.2978424`.

15 Ranjit Kumaresan, Tal Moran, and Iddo Bentov. How to use bitcoin to play decentralized poker. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 195–206, New York, NY, USA, 2015. Association for Computing Machinery. `doi:10.1145/2810103.2813712`.

16 Ranjit Kumaresan, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Improvements to secure computation with penalties. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 406–417, New York, NY, USA, 2016. Association for Computing Machinery. `doi:10.1145/2976749.2978421`.

17 Andrew Y. Lindell. Legally-enforceable fairness in secure two-party computation. In Tal Malkin, editor, *Topics in Cryptology – CT-RSA 2008*, pages 121–137, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

18 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*, March 2009.

19 Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

20 Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pages 162–167, USA, 1986. IEEE Computer Society. `doi:10.1109/SFCS.1986.25`.

$$P_1 \xrightarrow[q,\tau_n]{T_1\wedge\cdots\wedge T_n} P_n \qquad (1)$$

$$P_2 \xrightarrow[q,\tau_n]{T_1\wedge\cdots\wedge T_n} P_n \qquad (2)$$

$$\vdots$$

$$P_{n-1} \xrightarrow[q,\tau_n]{T_1\wedge\cdots\wedge T_n} P_n \qquad (n-1)$$

$$P_n \xrightarrow[(n-1)q,\tau_{n-1}]{T_1\wedge\cdots\wedge T_{n-1}} P_{n-1} \qquad (n)$$

$$P_{n-1} \xrightarrow[(n-2)q,\tau_{n-2}]{T_1\wedge\cdots\wedge T_{n-2}} P_{n-2} \qquad (n+1)$$

$$\vdots$$

$$P_4 \xrightarrow[3q,\tau_3]{T_1\wedge T_2\wedge T_3} P_3 \qquad (2n-4)$$

$$P_3 \xrightarrow[2q,\tau_2]{T_1\wedge T_2} P_2 \qquad (2n-3)$$

$$P_2 \xrightarrow[q,\tau_1]{T_1} P_1 \qquad (2n-2)$$

$$P_1 \xrightarrow[q,\tau_4]{T_1\wedge\cdots\wedge T_n} P_n \qquad (1)$$

$$P_2 \xrightarrow[q,\tau_4]{T_1\wedge\cdots\wedge T_n} P_n \qquad (2)$$

$$\vdots$$

$$P_{n-1} \xrightarrow[q,\tau_4]{T_1\wedge\cdots\wedge T_n} P_n \qquad (n-1)$$

$$P_n \xrightarrow[(n-1)q,\tau_3]{T_1\wedge\cdots\wedge T_{n-1}} P_{n-1} \qquad (n)$$

$$P_{n-1} \xrightarrow[(n-1)q,\tau_2]{T_{n-1}\wedge T_{n-2}} P_{n-2} \qquad (n+1)$$

$$P_{n-1} \xrightarrow[(n-1)q,\tau_2]{T_{n-1}\wedge T_{n-3}} P_{n-3} \qquad (n+2)$$

$$\vdots$$

$$P_{n-1} \xrightarrow[(n-1)q,\tau_2]{T_{n-1}\wedge T_1} P_1 \qquad (2n-2)$$

$$P_{n-2} \xrightarrow[(n-2)q,\tau_1]{T_{n-1}} P_{n-1} \qquad (2n-1)$$

$$P_{n-3} \xrightarrow[(n-2)q,\tau_1]{T_{n-1}} P_{n-1} \qquad (2n)$$

$$\vdots$$

$$P_1 \xrightarrow[(n-2)q,\tau_1]{T_{n-1}} P_{n-1} \qquad (3n-4)$$

Figure 1 Bentov–Kumaresan's fair reconstruction protocol (left) and our fair reconstruction protocol (right) in the n-party setting: In the deposit phase, the transactions are created from top to bottom, i.e., (1) to $(2n-2)$ in the left protocol and (1) to $(3n-4)$ in the right protocol. In the claim phase, the transactions are claimed in the reverse direction, i.e., $(2n-2)$ to (1) in the left protocol and $(3n-4)$ to (1) in the right protocol. The horizontal lines separate each round. Namely, in the deposit (resp. claim) phase, transactions belonging to the same section are created (resp. claimed) in one round.

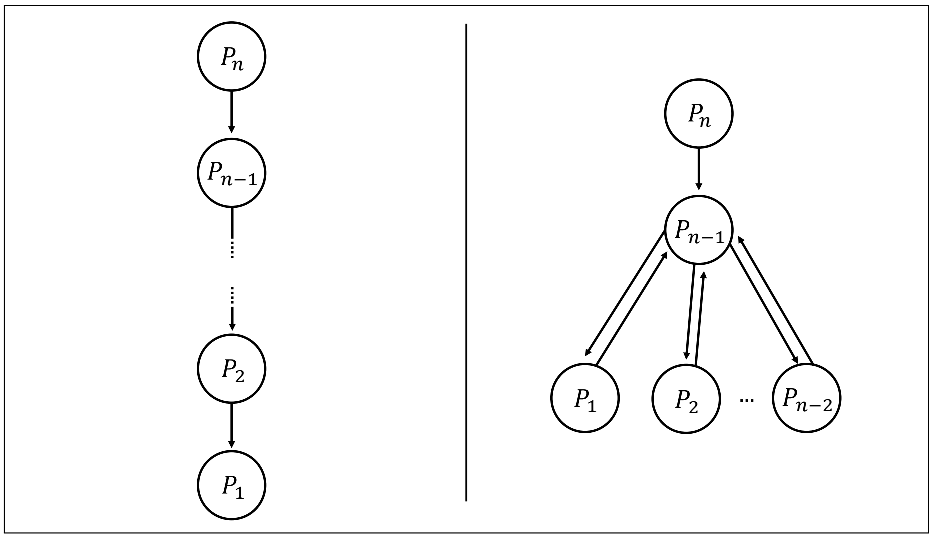


Figure 2 Flow of Bentov–Kumaresan's fair reconstruction protocol (left) and ours (right).

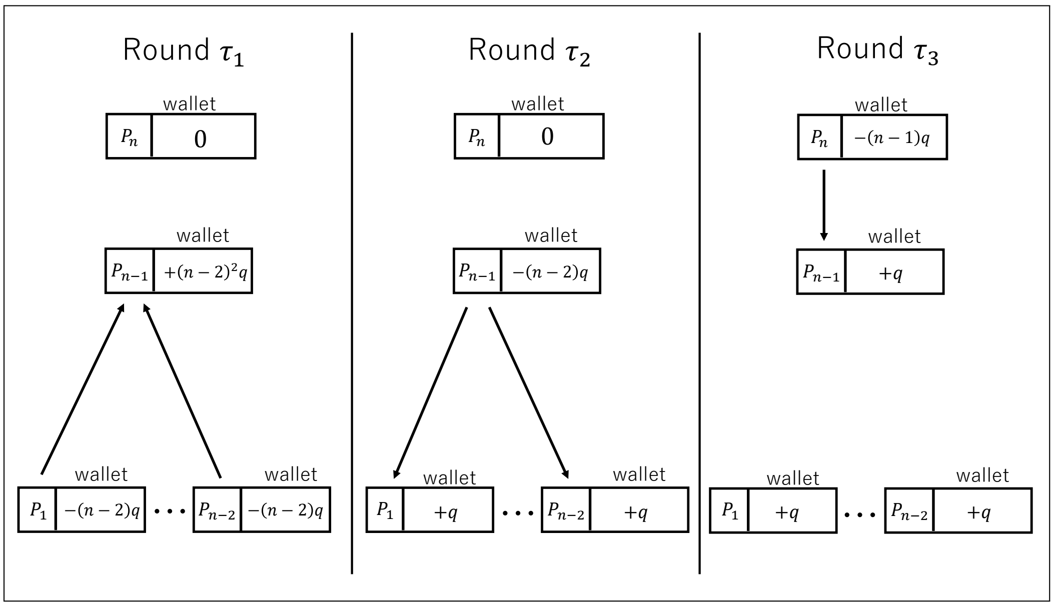


Figure 3 Coins flow in round τ_1 to τ_3 in the case where all parties behave honestly.

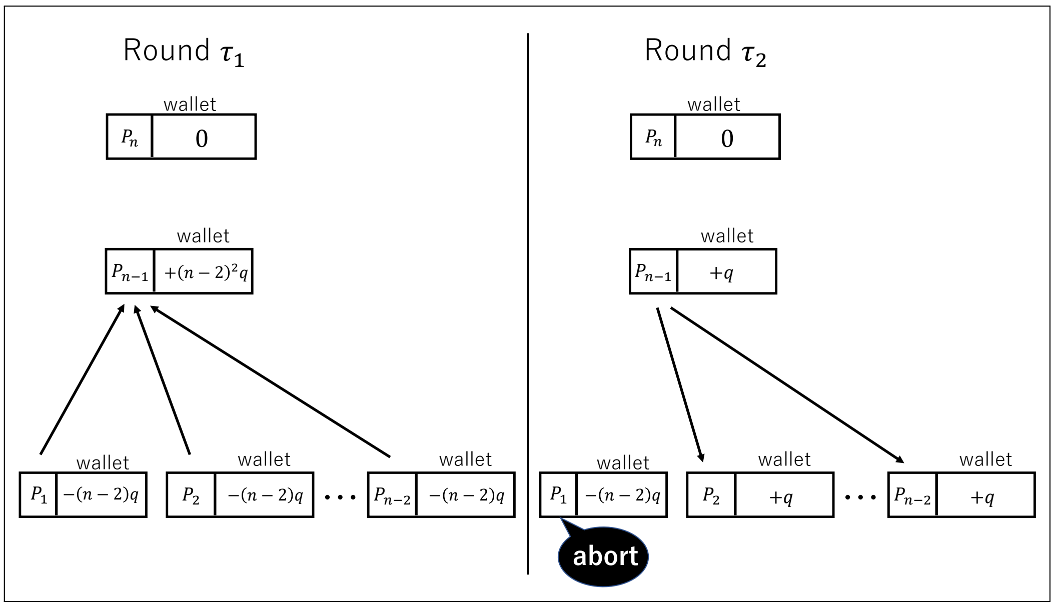


Figure 4 Coins flow in round τ_1 to τ_2 in the case where P_1 aborts.

Dynamic Posted-Price Mechanisms for the Blockchain Transaction Fee Market

Matheus V. X. Ferreira
Computer Science, Harvard University, Boston, MA, USA

Daniel J. Moroz
Computer Science, Harvard University, Boston, MA, USA

David C. Parkes
Computer Science, Harvard University, Boston, MA, USA

Mitchell Stern
EECS, University of California at Berkeley, CA, USA

Abstract

In recent years, prominent blockchain systems such as Bitcoin and Ethereum have experienced explosive growth in transaction volume, leading to frequent surges in demand for limited block space and causing transaction fees to fluctuate by orders of magnitude. Existing systems sell space using first-price auctions; however, users find it difficult to estimate how much they need to bid in order to get their transactions accepted onto the chain. If they bid too low, their transactions can have long confirmation times. If they bid too high, they pay larger fees than necessary.

In light of these issues, new transaction fee mechanisms have been proposed, most notably EIP-1559, aiming to provide better usability. EIP-1559 is a history-dependent mechanism that relies on block utilization to adjust a base fee. We propose an alternative design – *a dynamic posted-price mechanism* – which uses not only block utilization but also observable bids from past blocks to compute a posted price for subsequent blocks. We show its potential to reduce price volatility by providing examples for which the prices of EIP-1559 are unstable while the prices of the proposed mechanism are stable. More generally, whenever the demand for the blockchain stabilizes, we ask if our mechanism is able to converge to a stable state. Our main result provides sufficient conditions in a probabilistic setting for which the proposed mechanism is approximately welfare optimal and the prices are stable. Our main technical contribution towards establishing stability is an iterative algorithm that, given oracle access to a Lipschitz continuous and strictly concave function f, converges to a fixed point of f.

2012 ACM Subject Classification Theory of computation → Market equilibria; Security and privacy → Mathematical foundations of cryptography

Keywords and phrases Blockchain, Posted-price mechanism, Credible, Incentive compatibility, Transaction fee market, first-price auction, EIP-1559

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2021.6

Category Invited Talk

Related Version *Full Version*: `https://arxiv.org/abs/2103.14144` [1]

Acknowledgements The authors thank Cemil Dibek, Vikram V. Ramaswamy, and Tim Roughgarden for helpful discussions.

References

1 Matheus V. X. Ferreira, Daniel J. Moroz, David C. Parkes, and Mitchell Stern. Dynamic posted-price mechanisms for the blockchain transaction-fee market. In *AFT '21: 3rd ACM Conference on Advances in Financial Technologies*, pages 86–99. ACM, 2021.

Best Before? Expiring Central Bank Digital Currency and Loss Recovery

Charles M. Kahn
University of Illinois at Urbana-Champaign, IL, USA

Maarten R.C. van Oordt
Vrije Universiteit, Amsterdam, The Netherlands
Tinbergen Institute, Amsterdam, The Netherlands

Yu Zhu
Bank of Canada, Ottawa, ON, Canada

Abstract

An important feature of physical cash payments is resilience, due to their independence of power outages or network coverage. Many central banks are exploring issuing digital cash substitutes with similar offline payment functionality. Such substitutes could incorporate novel features making them more desirable than physical cash. This paper considers introducing an expiry date for offline digital currency balances to automate personal loss recovery. We show this functionality could increase consumer demand for digital cash, with the time to expiration playing a key role. The optimal time to expiration should have short wait time to get lost money reimbursed but leave enough time for agents to deposit received offline balances. Setting the time to expiration a bit too long has minor impact on welfare but setting it too short dramatically reduce welfare. If the offline device provides information about past transactions to the central bank, the accuracy of loss recovery can be improved but welfare can decrease.

2012 ACM Subject Classification Information systems → Digital cash

Keywords and phrases Central Bank Digital Currency, Design, Offline Payments, Operational Resilience, Financial Inclusion

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2021.7

Category Extended Abstract

Acknowledgements Views expressed do not necessarily represent official positions of the Bank of Canada.

Optimal Design of Tokenized Markets

Michael Junho Lee[1]
Federal Reserve Bank of New York, NY, USA

Antoine Martin
Federal Reserve Bank of New York, NY, USA

Robert M. Townsend
Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

Trades in today's financial system are inherently subject to settlement uncertainty. This paper explores tokenization as a potential technological solution. A token system, by enabling programmability of assets, can be designed to eradicate settlement uncertainty. We study the allocations achieved in a decentralized market with either the legacy settlement system or a token system. Tokenization can improve efficiency in markets subject to a limited commitment problem. However, it also materially alters the information environment, which in turn aggravates a hold-up problem. This limits potential gains from resolving settlement uncertainty, particularly for markets that depend on intermediaries.

2012 ACM Subject Classification Social and professional topics → Computing / technology policy; Computer systems organization → Distributed architectures

Keywords and phrases Tokenization, programmability, settlement uncertainty, asymmetric information

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2021.8

Related Version
Full Version: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3820973

[1] Corresponding author. The views expressed in this paper are those of the authors and do not necessarily reflect the position of the Federal Reserve Bank of New York or the Federal Reserve System.

To Infinity and Beyond: Financing Platforms with Uncapped Crypto Tokens

Rowena Gan
Information Technology and Operations Management Department, Cox School of Business, Southern Methodist University, Dallas, TX, USA

Gerry Tsoukalas
Information Systems Department, Questrom School of Business, Boston University, MA, USA

Serguei Netessine
Operations, Information and Decisions Department, The Wharton School, University of Pennsylvania, Philadelphia, PA, USA

Abstract

Initial Coin Offerings (ICOs) are an emerging form of crowdfunding for blockchain-based startups. While ICO design varies greatly in practice, many service-based platforms (e.g., Ethereum), use "uncapped" structures that forego limits on token supply, subjecting early investors to dilution risk. In this paper, we examine the conditions under which such ICOs are optimal and provide guidance for their design. Despite their popularity in practice, uncapped ICOs are understudied and not as well understood as their capped counterparts. We model game-theoretic interactions among various stakeholders in an infinite-horizon setting with network effects, taking account of operational details. We show that uncapped ICOs weakly dominate capped ones in the context of service platforms. In terms of design, a platform commission and regulation are generally "substitutes" when it comes to overcoming moral hazard, but can also be combined to make ICOs more accessible, especially for platforms with high initial setup costs. ICO accessibility can also be increased by employing a dual token offering (security & transaction tokens), at the cost of reduced expected profit. The paper provides a theoretical underpinning for the use of uncapped ICOs in practice. At a high level, it shows that ICOs succeed more easily in the presence of regulation, and platforms with low (high) setup costs should preferably issue utility (dual token) type ICOs.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Blockchain, Crowdfunding, Cryptocurrency, Dual Token Issuance, Initial Coin Offerings, ICOs, Moral Hazard, Stable Coins, Security Token Offerings, STOs

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2021.9

Category Extended Abstract

Detecting and Quantifying Crypto Wash Trading

Lin William Cong
Samuel Curtis Johnson Graduate School of Management,
Cornell University SC Johnson College of Business, Ithaca, NY, USA

Xi Li
Newcastle University Business School, UK

Ke Tang
Tsinghua University Institute of Economics, Beijing, China

Yang Yang
Tsinghua University Institute of Economics, Beijing, China

Abstract

We introduce systematic tests exploiting robust statistical and behavioral patterns in trading to detect fake transactions on 29 cryptocurrency exchanges. Regulated exchanges feature patterns consistently observed in financial markets and nature; abnormal first-significant-digit distributions, size rounding, and transaction tail distributions on unregulated exchanges reveal rampant manipulations unlikely driven by strategy or exchange heterogeneity. We quantify the wash trading on each unregulated exchange, which averaged over 70% of the reported volume. We further document how these fabricated volumes (trillions of dollars annually) improve exchange ranking, temporarily distort prices, and relate to exchange characteristics (e.g., age and userbase), market conditions, and regulation.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases Bitcoin, Cryptocurrency, FinTech, Forensic Finance, Fraud Detection, Regulation

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2021.10

Category Extended Abstract

Related Version *Full Version*: `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3530220`

Funding This research was funded in part by the Ewing Marion Kauffman Foundation and National Science Foundation of China, and the authors have no affiliation with or research support from any cryptocurrency exchanges.

Acknowledgements The authors are especially grateful to Deeksha Gupta, Kose John, Evgeny Lyandres, and Tao Li for repeated discussions and detailed feedback. We also thank Marlene Amstad, Mykola Babiak, Kevin Dowd, Valeria Ferrar, Itay Goldstein, Hanna Halaburda, Angel Hernando-Veciana, Andrew Karolyi, Dongyongp Lee, Minhyuk Lee, Jiasun Li, Laura Xiaolei Liu, Roger Loh, Emmanouil Platanakis, Fahad Saleh, Amin Shams, Donghwa Shin, Rajeev Singhal, Baolian Wang, Shang-jin Wei, Wei Xiong, Scott Yonker and seminar and conference participants and reviewers at the Alibaba Group Luohan Academy Webinar, Australasian Banking and Finance Conference, Behavioral Finance/Corporate Finance/Digital Finance (BF/DF/CF) Seminar Group, Cornell University, Cowles Foundation for Research In Economics Conference on the Economics of Cryptocurrencies, 11th CSBF Conference (National Taiwan University), Cowles Foundation Economics of Cryptocurrencies (Macroeconomics) Conference, 1st Crypto and Blockchain Economics Research Conference, Durham University Department of Economics and Finance, Econometric Society World Congress (Bocconi University), 2021 Eastern Finanace Association Annual Meeting, IIF International Research Conference & Award Summit, 13th International Risk Management Conference, Inaugural Machine Laywering Conference: "Human Sovereignty and Machine Efficiency in the Law," 18th Paris December Finance Meeting, Paris FinTech and Crypto Webinar, 60th Southwestern Finance Association Meeting, Sun Yat-sen University, 3rd Toronto FinTech Conference, Tsinghua University PBC School of Finance, 3rd UWA Blockchain and Cryptocurrency Conference, 11th Financial Markets and Corporate Governance Conference, European Financial Management Association Annual Meeting 2021, China International Conference in Finance 2021, 17th Asia-Pacific Association of Derivatives Annual Conference, World Finance Conference 2021, 2021 Financial Management Association Annual Meeting, 3rd International Conference on Blockchain Economics (Tokenomics2021), 2021 Global AI Finance Research Conference, Xi'an Jiaotong University, and the Zhongnan University of Economics and Law for helpful comments.

1 Crypto Wash Trading

The market capitalization of all cryptocurrencies exceeded 1.5 trillion USD in Feb 2021, and the total trading volume is 8.8 trillion USD in the first quarter of 2020 alone [6]. Both financial institutions and retail investors have significant exposure to the cryptocurrency industry.[1] Meanwhile, crypto exchanges, arguably the most profitable players in the ecosystem, remain mostly unregulated with less than 1% transactions taking place on regulated crypto exchanges in 2019 as we posted the first draft of the paper. In the process of vying for dominance in this lightly regulated market, some exchanges are suspected of gaining an advantage in ways ethically and legally questionable [11, 3]. One form of such market manipulation is Wash trading – investors simultaneously selling and buying the same financial assets to create artificial activity in the marketplace, which is known to distort price, volume, and volatility, and reduce investors' confidence and participation in financial markets [1].

Against such a backdrop, we conduct the first academic study of wash trading and misreporting on cryptocurrency exchanges. By inspecting the distribution of trade size whose first significant digit should follow Benford's law, should exhibit clustering at round numbers, and whose tail distribution is traditionally described by power law (Pareto-Levy law), we find that most unregulated exchanges wash trade (fabricating trades and acting as the counterparty on both sides to inflate volume).[2] We also estimate that unregulated exchanges on average inflate over 70% of the reported volumes in our sample. Furthermore, we provide suggestive evidence that the misreporting (generically referred to as wash trading) improves their ranking and prominence within the industry, relates to short-term price dispersion across exchanges, occurs more on newly established exchanges with smaller userbases, and has implications for the long-term industrial organization, development, and regulations.

While industry reports in 2018-2019 constitute whistle-blowers, their analyses are often imprecise, ad hoc, unscalable, and non-transparent [4]. Practitioners were unsure if wash trading only concerns a few specific exchanges with legal cases or was widespread; neither did they know how regulations play a role. Our goal is not to identify a specific wash trade, but to rigorously establish that wash trading is a rampant, industry-wide issue for the cryptocurrency market. We are among the earliest to provide evidence for the efficacy

[1] Surveys reveal that 22% institutional investors have invested in cryptocurrencies [10] and by April 2019 9% of adults have owned Bitcoins in particular [2]. In the UK, 25% consumers could identify "cryptocurrency" and 3% had bought them [5]. Between 2016 and 2018, Bitcoin ownership increased from 3% to 5% [7]

[2] Wash trading is, according to the U.S. Commodity Exchange Act, "Entering into, or purporting to enter into, transactions to give the appearance that purchases and sales have been made, without incurring market risk or changing the trader's market position." Definition of wash trading from US Commodity Exchange Act can be found at `https://www.cftc.gov/ConsumerProtection/EducationCenter/CFTCGlossary/glossary_wxyz.html`

of regulation in this industry, which has implications for investor protection and financial stability.[3] Our findings also likely have consequences for ongoing lawsuits and empirical research on cryptocurrencies which frequently reference transaction volumes. Finally, they serve as illustrations of the usefulness of statistical and behavioral principles for forensic finance, with regulatory implications for FinTech and beyond.

Wash trading on crypto exchanges warrants our attention for several reasons. First, crypto exchanges play essential roles in the industry, providing liquidity and facilitating price discovery just like traditional exchanges. Many crypto exchanges have expanded into upstream (e.g., mining) and downstream (e.g., payment) sectors, consequently wielding great influence as a complex of trading platforms, custodians, banks, and clearinghouses. Naturally, crypto exchanges constitute an anchoring point for understanding the ecosystem from academic, industrial, and regulatory perspectives. Second, because liquidity begets liquidity, crypto exchanges have strong economic incentives to inflate trading volumes to increase brand awareness and ranks on third-party aggregator websites or media (e.g., CoinMarketCap, CoinGecko, Bitcointalk, and Reddit), which in turn increases the exchanges' profits from transaction fees. Third, while wash trading is largely prohibited in most financial markets and developed economies [8], cryptocurrencies are particularly prone to wash trading, under limited regulatory oversight.

We collect cryptocurrency transaction information on 29 major exchanges from the proprietary database maintained by TokenInsight (www.tokeninsight.com), a data provider who offers consulting, rating, and research reports for the cryptocurrency-related business. We adopt the definition of regulated exchanges from the state of New York, which has one of the earliest regulatory frameworks in the world.[4] We then use web traffic ranking as a proxy for brand awareness and reputation to further categorize unregulated exchanges for easy reference: "Tier-1" for exchanges ranking in the top 700 in the finance/investment section of SimilarWeb.com and "Tier-2" for the rest of unregulated exchanges on our data (all ranking outside top 960). Our data cover the period from 00:00 July 09th, 2019 (when TokenInsight started to collect transaction information from these exchanges) to 23:59 November 03rd, 2019 (the time we wrote the first draft). Our data also contain variables including aggregate trading volume, reputation metrics, and exchange characteristics such as exchange age. For each exchange, we examine several most widely recognized and heavily traded cryptocurrencies against US dollars (USD), including Bitcoin (BTC), Ethereum (ETH), Litecoin (LTC), and Ripple (XRP).

We are fully aware of the challenges of forensic finance and employ multiple approaches that have been successfully applied in numerous fields in sciences and social sciences and are shown to be unlikely affected by dispersed traders' strategies, exchange characteristics, or specificities of the asset class. To start, we examine the first significant digit for each

[3] After we publicly circulated our study in late 2019, several ranking websites changed their matrices from purely volume-based to more sophisticated multi-dimensional ranking models, with at least one website doing so in response to our research finding. Regulators also increased scrutiny on wash trading behavior: Canada-based crypto trading platform Coinsquare has agreed to settle with the Ontario Securities Commission for wash trading charge; Coinbase was fined before IPO for wash trading several years earlier.

[4] Regulated exchanges are issued BitLicenses and are regulated by the New York State Department of Financial Services. Bitlicence carries some of the most stringent requirements. Our main results are robust to alternative classifications of regulated exchanges. As of June 2020, NYDFS has issued licenses to 25 regulated entities, six of which provide crypto exchange service. They are Itbit, Coinbase, Bitstamp, Bitflyer, Gemini, and Bakkt (futures and options only). Further information can be found at: `https://www.dfs.ny.gov/apps_and_licensing/virtual_currency_businesses/regulated_entities`. (Last accessed: July 3, 2020)

transaction and check its frequency distribution on each exchange against Benford's law – the well-known statistical benchmark in natural sciences and social sciences and widely used to detect frauds in macroeconomic, accounting and engineering fields. We next exploit a classical behavioral regularity in trading: clustering at certain transaction sizes. Round numbers are routinely used as cognitive reference points in individuals' decision-making. Rounding is commonly observed in finance, including analysts' forecasts or LIBOR submissions. Our third test explores whether the distributions of observed trade size have fat tails characterized by the power law as seen in traditional financial markets and other economic settings. We fit a power-law distribution and estimate the exponent parameters in addition to graphically inspecting the tail distributions on a log-log scale. We consistently find anomalous trading patterns only on unregulated exchanges, with Tier-1 exchanges failing more than 20% of the tests and Tier-2 exchanges failing more than 60%. The findings remain robust when under joint hypothesis tests.

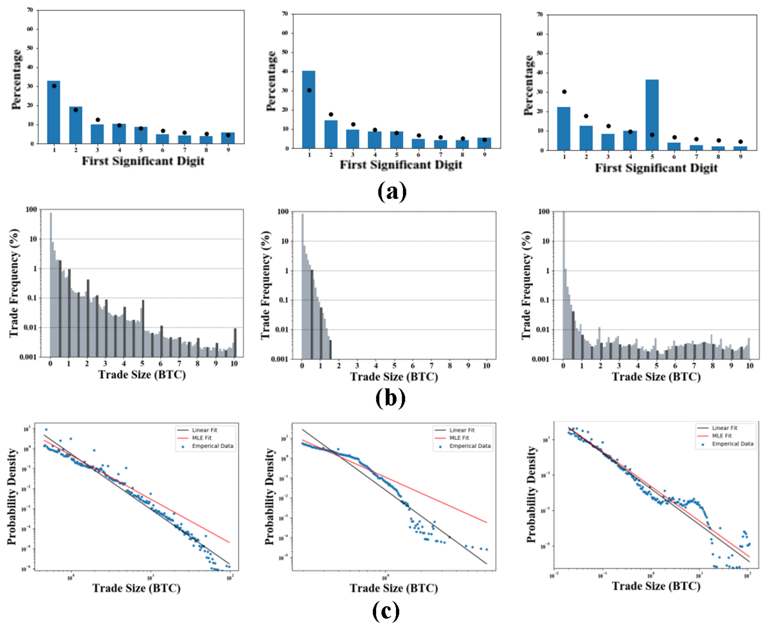


Figure 1 First-significant-digit, Trade-size Clustering, and Tail Distribution of Trade Size. The figure demonstrates the BTC/USD distribution drawn from sample exchanges compared with the benchmarks. Three exchanges are shown in the figure, one from each category (regulated exchange-left, Tier 1 unregulated exchanges-center, Tier 2 unregulated exchanges-right). Panel (a) are the first-significant-digit distributions and comparisons with Benford's law. Black dots represent distributions derived from Benford's law. Distributions of trading data are reported in bar charts. Panel (b) shows the clustering effect in trade-size distributions histograms on exchanges. In each histogram, we highlight every 5th and 10th bin to illustrate the clustering effect around round trade sizes. Panel (c) displays tails of trade-size distributions and the fitted power-law lines on log-log plots. Fitted power-law lines are plotted with parameters estimated by Ordinary Least Square (OLS) and Maximum Likelihood Estimation (MLE), shown in black and red lines, respectively. Blue dots represent empirical data points for trade-size frequencies.

As Figure 1 shows, the example unregulated exchanges deviate from all three benchmarks, Benford's Law, clustering effect, and power law, significantly. This holds for all four currency pairs on majority unregulated exchanges. While current business incentives and ranking systems fuel the rampant wash trading on unregulated exchanges, the regulated exchanges, having committed considerable resources towards compliance and license acquisition and facing severe punishments for market manipulation [9], conduct little wash trading.

Besides identifying exchanges that wash trade, we quantify the fractions of fake volume by taking advantage of the rounding regularity. A benchmark ratio (based on calculations from the regulated exchanges) of unrounded trades to authentic trades with round sizes are calculated. The extra unrounded trades above the ratio naturally constitute wash trades on unregulated exchanges. We find that the wash trading volume on average is as high as 77.5% of the total trading volume on the unregulated exchanges, with a median of 79.1%. In particular, wash trades on the twelve Tier-2 exchanges are estimated to be more than 80% of the total trade volume, which is still over 70% after accounting for observable exchange heterogeneity. Combined with the reported volumes in Helms' article [6], these estimates translate into wash trading of over 4.5 Trillion USD in spot markets and over 1.5 Trillion USD in derivatives markets in the first quarter of 2020 alone. To mitigate the influence of heterogeneity of traders and algorithmic trading strategies across various exchanges, we validate the roundness-ratio estimation and conduct a number of robustness tests to allay selection concerns.

Table 1 Aggregated Wash Trading Percentage. The table presents the simple averaged and volume-weighted wash trading percentage for each exchange category.

	Wash Trade Percentage Without Control Variables		Wash Trade Percentage With Control Variables	
	Equal-weighted Average	Volume-weighted Average	Equal-weighted Average	Volume-weighted Average
Unregulated	70.85	77.50	60.96	71.43
Unregulated Tier-1	53.41	61.86	46.95	63.62
Unregulated Tier-2	81.76	86.26	70.96	76.96

We then study exchange characteristics that correlate with wash trading and investigate the impact of wash trading on market outcomes such as exchange ranking. In addition, we obtain proprietary data on historical ranking and trading volume information from CoinMarketCap and show that exchange ranking depends on wash trading (70% wash trading of total reported volume moves an exchange's rank up by 46 positions). We find that an exchange's wash trading is positively correlated with its cryptocurrency prices over the short term. We also find that exchanges with longer establishment history and larger userbase wash trade less. Less prominent exchanges, in contrast, have short-term incentives for wash trading without drawing too much attention. Moreover, wash trading is positively predicted by returns and negatively by price volatility.

To sum up, as the first comprehensive study of the pervasive crypto wash trading, our paper not only provides a cautionary tale to regulators around the globe but also reminds the readers of the disciplining or screening effects of regulation in emerging industries, the importance of using wash-trading-adjusted volume in certain empirical studies, and the utility of statistical tools and behavioral benchmarks for forensic finance and fraud detection. We offer a concrete set of tools for exchange regulation and third-party supervision in the crypto market for convincingly exposing wash trading and potentially combating non-compliant exchanges. Admittedly, the tests we introduce are not exhaustive, and wash traders may adjust their strategies in response to these tests. Our tools nevertheless serve as valid detection of wash trading historically and thus make fabrications more difficult and facilitate regulatory resource allocation.

References

1 Rajesh Aggarwal and Guojun Wu. Stock market manipulations. *The Journal of Business*, 79(4):1915–1954, 2006. URL: `https://EconPapers.repec.org/RePEc:ucp:jnlbus:v:79:y:2006:i:4:p:1915-1954`.

2 Spencer Bogart. Blockchain capital bitcoin survey. *Blockchain Capital Blog*, 2019. URL: `https://medium.com/blockchain-capital-blog/bitcoin-is-a-demographic-mega-trend-data-analysis-160d2f7731e5?`

3 BTI. April summary of market surveillance report. Technical report, Blockchain Transparency Institute, 2019. URL: `https://www.bti.live/reports-april2019/`.

4 Sead Fadilpasic. Okex defends itself from wash trading accusations with a btc 100 bet. *CryptoNews*, 2019. URL: `https://cryptonews.com/news/okex-defends-itself-from-wash-trading-accusations-with-a-btc-4720.htm`.

5 FCA. Cryptoassets: Ownership and attitudes in the uk. Technical report, Financial Conduct Authority, 2019. URL: `https://www.fca.org.uk/publication/research/cryptoassets-ownership-attitudes-uk-consumer-survey-research-report.pdf`.

6 Kevin Helms. $8.8 trillion traded in cryptocurrency spot and futures markets in q1: Reports. *Bitcoin.com News*, 2020. URL: `https://news.bitcoin.com/trillion-traded-cryptocurrency-spot-futures-markets/`.

7 Christopher S Henry, Kim P Huynh, and Gradon Nicholls. Bitcoin awareness and usage in canada: An update. *The Journal of Investing*, 28(3):21–31, 2019.

8 IOSCO. A resolution on iasc standards. *Presidents' Committee of IOSCO, Spain*, 2000.

9 Yessi B Perez. The real cost of applying for a new york bitlicense. *Coindesk*, 2015. URL: `https://www.coindesk.com/real-cost-applying-new-york-bitlicense`.

10 Arlene Roberts. Institutional investments in digital assets. Technical report, Fidelity, 2019. URL: `https://s2.q4cdn.com/997146844/files/doc_news/archive/59439969-390c-4354-94a9-772219d0b8b9.pdf`.

11 Paul Vigna. Most bitcoin trading faked by unregulated exchanges, study finds. *The Wall Street Journal*, 2019. URL: `https://www.wsj.com/articles/most-bitcoin-trading-faked-by-unregulated-exchanges-study-finds-11553259600`.

Fundamentals of the MakerDAO Governance Token

Roman Kozhan
Warwick Business School, University of Warwick, Coventry, UK

Ganesh Viswanath-Natraj
Warwick Business School, University of Warwick, Coventry, UK

Abstract

We study the fundamentals governing the price of the MakerDAO governance token MKR. Governance tokens are minted in response to liquidations, and burned in response to growth in the system surplus. MKR tokens appreciate with an increase in system surplus and depreciate with a rise in systemic risk due to DAI liquidation spirals. We discuss incentive compatibility conditions that need to be satisfied for the protocol to maintain the DAI stablecoin peg.

2012 ACM Subject Classification Applied computing

Keywords and phrases Cryptocurrency, monetary policy, stablecoins, governance token

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2021.11

1 Introduction and Motivation

Decentralized stablecoins are led by MakerDAO's DAI token, in which the issuance of new tokens is decentralized through using autonomous smart contracts on the Ethereum blockchain.[1] DAI tokens are generated when an investor deposits a set amount of collateral, typically Ethereum (ETH), into a collateralized debt position (CDP).

Empirical work on DAI has focused on understanding the role of collateral risk and peg stability, the fundamental sources of peg-price deviations and demand for DAI in DeFi applications. [5, 2, 3, 7]. Theoretical contributions of DAI include a study of liquidations, the incentive compatibility of governance structures, and the use of a reserve buffer as a solution to peg instability [4, 6, 1].

In this paper, we discuss the fundamentals of the governance token price, the supply mechanism of mints and burns, and the incentive compatibility conditions that need to be met for peg stability. In our empirical analysis, we find the MKR governance token appreciates when MKR tokens are burned due to growth in the system surplus. MKR tokens are minted in response to debt auctions to cover losses on liquidating DAI loans. Finally, we discuss the role of the voting mechanism used in governance. The governance protocol can lead to inefficient outcomes if MKR token holdings are concentrated with a single investor. The remainder of the paper is structured as follows. In section 2 we outline fundamentals of the MKR token. Section 3 discusses issues in mechanism design and whether the governance protocol is incentive compatible. Section 4 concludes.

[1] A smart contract is a set of instructions, written in computer code, that defines the conditions of the contract for each counterparty under different scenarios (default etc). Being managed by computer code and visible on the blockchain, it can be verified publicly by all nodes in the blockchain.

2 Fundamentals of Governance Token Valuation

The Maker Governance protocol is in charge of adding new collateral types, the regulation of the smart contracts enforcing collateralize debt positions, and adjusting risk parameters of the protocol, such as the liquidation ratio, debt ceilings and the stability and savings rate. Panel A of Figure 1 illustrates the MKR price. In panel B, the MakerDAO token launched with a supply of 1 million MKR. In panel C, we plot the net redemptions of MKR tokens. In panel D, the system surplus is the amount of DAI generated from system fees, including Stability Fees and Liquidation Fees set by Maker governance. In panel E, we plot the ratio of the MKR price to the system surplus per unit of the MKR token.[2] In Panel F, we show that Systemic risk in ETH is strongly correlated with volatility in MKR. We now outline our three hypotheses of fundamentals of MKR token valuation.

▶ **Hypothesis 1.** *System surplus* $\uparrow \implies$ *MKR tokens burned* $\implies$ *MKR prices* $\uparrow$

When the system surplus exceeds the safety buffer, any additional DAI is auctioned off for MKR, the governance token of the Maker Protocol, in lots of 10,000 DAI in a Surplus Auction. The system then burns the MKR it receives in the Surplus Auction, reducing the total supply. In 2021, strong growth in the system surplus due to stability and liquidation fees has led to a net reduction of MKR tokens through surplus auctions. We hypothesize that growth in the system surplus leads to a net burning of MKR tokens and an increase in the valuation of the MKR token.

▶ **Hypothesis 2.** *Liquidations* $\uparrow \implies$ *MKR tokens minted* $\implies$ *MKR prices* $\downarrow$

During the *Black Thursday Crypto crash* on March 12th 2020, MKR tokens were minted to pay off the DAI debt triggered by liquidations. If the sale of collateral is not sufficient to pay off the DAI loans triggered in liquidation, the Protocol triggers a MKR Debt Auction. MKR is minted by the system, increasing the amount of MKR in circulation, and then sold to bidders for DAI. We hypothesize that the minting of MKR tokens devalues the MKR token.

▶ **Hypothesis 3.** *Volatility of ETH* $\uparrow \implies$ *Volatility of MKR* $\uparrow$ *and MKR prices* $\downarrow$

ETH volatility can translate to MKR volatility through (i) an increase in gas fees and congestion on the blockchain, (ii) more volatile demand for ERC-20 tokens and decentralized finance applications, and (iii) increased systemic risk of vaults that hold ETH collateral. We empirically test the three fundamentals of MKR prices in Equation (1):

$$Y_t = \beta_0 + \beta_1 \frac{\Delta Surplus_{t-1}}{MKR} + \beta_2 \sigma_{ETH,t} + \beta_3 Liquidation_t + u_t \tag{1}$$

Here, the outcome variable Y_t is the MKR return and the intra-day volatility. Intra-day volatility is calculated as the square root of the average sum of squared hourly returns over the trading day. Explanatory variables include the growth in the system surplus normalized by MKR tokens in circulation, ETH intra-day volatility, and liquidations (millions DAI).

[2] The valuation of the MKR token is analogous to a dividend that is paid to MKR stakeholders for supporting the governance protocol in maintaining the DAI peg.The valuation of a MKR token per unit of system surplus fluctuates between 100 to 500.

The results are summarized in Table 1. Consistent with our proposed hypotheses, MKR returns correlate positively with system surplus and negatively with liquidations. In column (IV), a 1 per cent increase in the ratio of system surplus to MKR tokens results in a 4.7 basis point increase in MKR returns, and a 1 million increase in DAI liquidations is associated with a 1.07 per cent decline in MKR returns. MKR volatility correlates significantly with ETH volatility with an estimated elasticity of 0.82 in column (VIII).

3 Governance and Incentive Compatibility Conditions

The MKR governance token is used for voting on the management of the protocol and setting many of the parameters of the DAI stablecoin peg. For example, to change the stability rate, each user places a vote on their preferred stability rate by staking their MKR tokens. Each MKR token equals one vote when locked in a voting contract. Users commit their Maker tokens to a proposal, with the outcome being decided by the number of MKR tokens it receives.

A rational user has the following objective function: to maximize the system surplus subject to the constraint of the DAI peg stability. Setting parameters that move the DAI price away from the peg can have adverse effects on system surplus if it causes DAI users to switch toward other stablecoins. For illustration, suppose that MKR holders decide to set a high debt ceiling for risky collateral types like ETH, and lower the debt ceiling for safe collateral types like USDC. While this policy change may increase short-run revenues, this increases systemic risk due to negative valuations of ETH. Therefore MKR holders need to internalize the setting of policy parameters impact on the DAI peg and to set parameters to minimize systemic risk to the protocol.

Governance voting protocols can lead to inefficient outcomes if tokens are concentrated with one user. An example is the vote on the stability fee on October 28th, 2019, in which an individual MKR holder staked up to 94% of the vote to change the stability fee from 9 % to 5%. [3] Such outcomes can be sub-optimal if they conflict with the target of peg stability. A concentration of MKR tokens with one user can also lead to inefficient prices of MKR auctions. The MakerDAO system burns MKR tokens when the system surplus exceeds the reserve buffer. The burning of MKR tokens takes place in an auction when users make bids for DAI tokens in return for selling their MKR tokens. If all MKR tokens are concentrated with one investor, they will be able to sell their MKR tokens at a high valuation, leading to significant losses for the protocol.

4 Conclusion

In this paper, we discuss the fundamentals of the governance token price and the incentive compatibility conditions that need to be met for peg stability. The MKR token typically appreciates in response to an increase in the system surplus, and depreciates when MKR tokens are minted in response to MKR debt auctions and to cover losses on liquidating DAI loans. While MKR holders have an incentive to set parameters to minimize systemic risk to the DAI peg and governance protocol, there are potential vulnerabilities when there is concentration of MKR tokens. A potential solution to the governance protocol is to set caps on the MKR staked to prevent a single user from being able to alter the outcome of policy parameters.

[3] `https://cryptoslate.com/makerdao-whale-with-94-voting-power-reduces-dai-stability-fee-by-4/`

Table 1 MKR Price Fundamentals: Liquidations.

	I	II	III	IV	V	VI	VII	VIII
	R_{MKR}	R_{MKR}	R_{MKR}	R_{MKR}	σ_{MKR}	σ_{MKR}	σ_{MKR}	σ_{MKR}
Surplus	0.0442***			0.0474***	0.0255***			0.0211***
	(0.0126)			(0.0124)	(0.0080)			(0.0069)
σ_{ETH}		-0.2828***		-0.1760		0.8285***		0.8283***
		(0.1049)		(0.1108)		(0.0573)		(0.0612)
Liquidation			-1.2039***	-1.0691***			0.9301***	-0.0671
			(0.3297)	(0.3504)			(0.2088)	(0.1936)
Intercept	0.2581	1.7793***	0.6902**	1.1143**	6.2645***	2.8575***	6.3346***	2.7228***
	(0.3265)	(0.5528)	(0.3170)	(0.5642)	(0.2083)	(0.3018)	(0.2007)	(0.3117)
R-squared	0.02	0.01	0.02	0.05	0.02	0.27	0.03	0.29
No. observations	555	555	555	555	555	555	555	555

Note: This table regresses MKR returns and volatility on ETH volatility,system surplus and liquidations. The dependent variable R_{MKR} measures the daily MKR return. The dependent variable σ_{MKR} measures the intra-day volatility of MKR prices. The sample runs from November 18th, 2019 to March 31st, 2021, corresponding to the period of Multi Collateral DAI. White heteroscedasticity-robust standard errors are reported in parentheses. *** denotes significance at the 1 percent level, ** at the 5 percent level, and * at the 10 percent level.

References

1 Adrien d'Avernas, Thomas Bourany, and Quentin Vandeweyer. Are stablecoins stable? *Working Paper*, 2021.

2 Barry Eichengreen. From commodity to fiat and now to crypto: What does history tell us? Technical report, National Bureau of Economic Research, 2019.

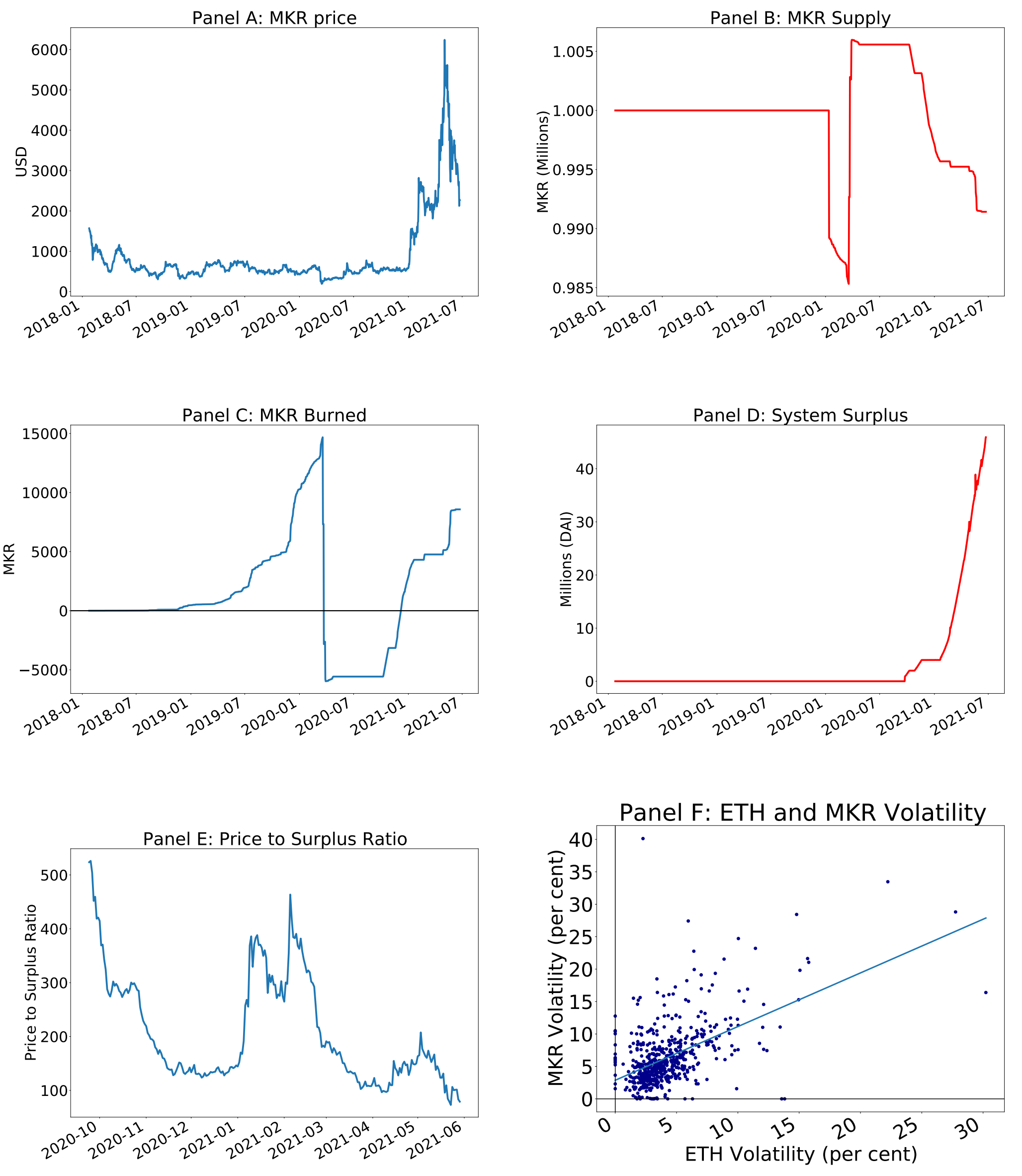
3 Klaudia Jarno and Hanna Kołodziejczyk. Does the design of stablecoins impact their volatility? *Journal of Risk and Financial Management*, 14(2):42, 2021.

4 Ariah Klages-Mundt and Andreea Minca. While stability lasts: A stochastic model of stablecoins. *arXiv preprint*, 2020. `arXiv:2004.01304`.

5 Roman Kozhan and Ganesh Viswanath-Natraj. Decentralized stablecoins and collateral risk. *WBS Finance Group Research Paper Forthcoming*, 2021.

6 Ye Li and Simon Mayer. Money creation in decentralized finance: A dynamic model of stablecoin and crypto shadow banking. *CESifo Working Paper*, 2021.

7 Kanis Saengchote. Where do defi stablecoins go? a closer look at what defi composability really means. *A closer look at what DeFi composability really means.(July 26, 2021)*, 2021.



Note: This figure plots Panel A: MKR price, Panel B: MKR Supply Panel C: MKR Burned, Panel D: System Surplus, Panel E: Ratio of MKR Price to Unit Surplus, Panel F: ETH and MKR Volatility. Sample period is from April 13th, 2018 to March 31st, 2020.

Figure 1 MKR price, MKR Supply, Burned/Minted Tokens and System Surplus.

Blockchain and Privacy

Catherine Tucker
MIT Sloan, MIT, Cambridge, MA, USA

Abstract

The unique value proposition of 'smart contracts' based on blockchain technology is the creation of a permanent public record of agreed-upon transactions that cannot be changed retroactively. Though this is attractive in terms of reducing the potential for fraud, a person entering into a smart contract pre-commits both their current self and their future selves, no matter what changes occur to them or to their circumstances. The advantages of such pre-commitments can be substantial, but even in an age of increasing adoption of distributed ledger technologies, self-reinvention remains important. From a surveillance perspective, it is important to prevent governments from reliably associating a particular cryptoasset transaction with a particular person. For individuals, it is important to preserve the ability to assume new identities both formally and informally. This presentation will present an expanded and refined understanding of what it means for a blockchain use case to "protect privacy," and in particular, how such use cases can encourage a notion of personal identity that is inflexible and matches poorly with individuals' notions regarding their identities. In addition I discuss how privacy regulation may itself shape the development of blockchain.

2012 ACM Subject Classification Security and privacy → Block and stream ciphers

Keywords and phrases Blockchain, Privacy

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2021.12

Category Invited Talk

Acknowledgements I want to thank Alex Marthews for coauthoring the paper with me that inspired this talk.

Presentation and Publication: Loss and Slippage in Networks of Automated Market Makers

Daniel Engel
Computer Science Department, Brown University, Providence, RI, USA

Maurice Herlihy
Computer Science Department, Brown University, Providence, RI, USA

Abstract

Automated market makers (AMMs) are smart contracts that automatically trade electronic assets according to a mathematical formula. This paper investigates how an AMM's formula affects the interests of liquidity providers, who endow the AMM with assets, and traders, who exchange one asset for another at the AMM's rates. *Linear slippage* measures how a trade's size affects the trader's return, *angular slippage* measures how a trade's size affects the subsequent market price, *divergence loss* measures the opportunity cost of providers' investments, and *load* balances the costs to traders and providers. We give formal definitions for these costs, show that they obey certain conservation laws: these costs can be shifted around but never fully eliminated. We analyze how these costs behave under *composition*, when simple individual AMMs are linked to form more complex networks of AMMs.

2012 ACM Subject Classification Theory of computation → Algorithmic mechanism design

Keywords and phrases Decentralized Finance, AMM, Uniswap

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2021.13

Funding *Daniel Engel*: This research was supported by NSF grant 1917990.
Maurice Herlihy: This research was supported by NSF grant 1917990.

1 Introduction

An *automated market maker* (AMM) is an automaton that trades electronic assets according to a fixed formula. Unlike traditional "order book" traders, AMMs have custody of their own asset pools, so they can trade directly with clients, and do not need to match up (and wait for) compatible buyers and sellers. Today, AMMs such as Uniswap [5], Bancor [17], and others have become one of the most popular ways to trade electronic assets such as cryptocurrencies, electronic securities, or tokens. An AMM is typically implemented as a smart contract on a blockchain such as Ethereum [13]. Like circuit elements, AMMs can be *composed* into networks. They can be composed sequentially, where the output of one AMM's trade is fed to another, and they can be composed in parallel, where a trade is split between two AMMs with different formulas. Compositions of AMMs can themselves be treated as AMMs [12]. This paper makes the following contributions. AMMs have well-known inherent costs. One such cost is *slippage*, where a large trade increases the price of the asset being purchased, both for the trader making the trade, and for later traders. We give two alternative mathematical definitions of slippage expressed directly in terms of the AMM's formula: *linear slippage* focuses on the buyer's price difference, and *angular slippage* characterizes how that buyer affects prices for later buyers.

Another cost is *divergence loss* (sometimes called *impermanent loss*), where the value of the liquidity providers' investments end up worth less than if the invested assets had been left untouched. We give a precise mathematical definition of divergence loss, expressed in directly terms of the AMM's formula, We introduce a new figure of merit, called *load*, that measures how costs are balanced among parties on different sides of a trade.

We show how these costs can be analyzed in either worst-case or in expectation. We identify various *conservation laws* that govern these costs: they can be shifted, but never fully eliminated. We characterize how these costs behave under sequential and parallel *composition*, showing how to compute these costs for networks of AMMs, not just individual AMMs. Finally, we propose novel AMM designs capable of adapting to changes in these costs.

The paper is organized as follows. Section 2 describes our model and terminology. Section 3 introduces our cost measures and their conservation laws. Section 4 shows how these measure are affected by sequential composition, where the output of one AMM becomes the input of another AMM. Section 4 shows how these measure are affected by parallel composition, where traders split their trade between two AMMs that trade the same assets but according to different formulas. Section 6 surveys some simple adaptive strategies that can mitigate the costs' conservation laws. Section 7 surveys related work.

Some of our numbered equations require long, mostly routine derivations which have been moved to the appendix to save space. A few of the longer proofs have also been moved to the appendix.

2 Definitions

We use bold face for vectors ($\boldsymbol{x}$) and italics for scalars (x). Variables, scalar or vector, are usually taken from the end of the alphabet (x, y, z), and constants from the beginning (a, b, c). We use ":=" for definitions and "=" for equality. A function $f : \mathbb{R} \to \mathbb{R}$ is *strictly convex* if for all $t \in (0, 1)$ and distinct $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}$, $f(t\boldsymbol{x} + (1-t)\boldsymbol{x}') < tf(\boldsymbol{x}) + (1-t)f(\boldsymbol{x}')$. Any tangent line for a strictly convex function lies below its curve.

Here is an informal example of a *constant-product* AMM [20]. An AMM in state (x, y) has custody x units of asset X, and y units of asset Y, subject to the invariant that the product $xy = c$, for $x, y > 0$ and some constant $c > 0$. The AMM's states thus lie on the hyperbolic curve $xy = c$. If a trader transfers δ_X units of X to the AMM, the AMM will return δ_Y units of Y to the trader, where δ_Y is chosen to preserve the invariant $(x + \delta_X)(y - \delta_Y) = c$.

Formally, the state of an AMM that trades assets X and Y is given by a pair $(x, y) \in \mathbb{R}^2_{>0}$, where x is the number of X units in the AMM's pool, and y the number of Y units. The state space is given by a curve $(x, f(x))$, where $f : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$. Except when noted, the AMMs considered here satisfy the boundary conditions $\lim_{x\to 0} f(x) = \infty$ and $\lim_{x\to\infty} f(x) = 0$, meaning that traders cannot exhaust either pool of assets. The function f is subject to further restrictions discussed later.

There are two kinds of participants in decentralized finance. (1) *Traders* transfer assets to AMMs, and receive assets back. Traders can compose AMMs into networks to conduct more complicated trades involving multiple kinds of assets. (2) *Liquidity providers* (or "providers") fund the AMMs by lending assets, and receiving shares, fees, or other profits. Traders and providers play a kind of alternating game: traders modify AMM states by trading one asset for another, and providers can respond by adding or removing assets, reinvesting fees, or adjusting other AMM properties.

AMM typically charge fees for trades. For example, Uniswap v1 diverts 0.3% of the assets returned by each trade back into that asset's pool. Although there is no formal difficulty including fees in our analysis, we neglect them here because they have little impact on costs: fees slightly reduce both slippage costs for traders and divergence loss for providers.

A *valuation* $v \in (0,1)$ assigns relative values to an AMM's assets: v units of X are deemed worth $(1-v)$ units of Y. At valuation v, a trader who moves an AMM from state $(x, f(x))$ to $(x', f(x'))$ makes a profit if $v(x - x') + (1-v)(f(x) - f(x'))$ is positive, and otherwise incurs a loss. The trader's profit is maximal precisely when $vx' + (1-v)f(x')$ is minimal. We assume that at any time, there is a single *market valuation* accepted by most traders. An *arbitrage trade* is one in which a trader makes a profit by moving an AMM from a state reflecting a prior market valuation to a distinct state reflecting the current market valuation.

A *stable point* for an AMM A and valuation v is a point $(x, f(x))$ that minimizes $vx + (1-v)f(x)$. If v is the market valuation, then any trader can make an arbitrage profit by moving the AMM from any state to a stable state, and no trader can make a profit by moving the AMM out of a stable state. Valuations, stable points, and exchange rates are related. If $(x, f(x))$ is the stable point for valuation v, then

$$\frac{df(x)}{dx} = -\frac{v}{1-v}. \tag{1}$$

Following Engel and Herlihy [12], we require the AMM function f to satisfy certain reasonable properties, expressed here as axioms. A detailed discussion and justification for each axiom appears elsewhere [12].

Every AMM state should define a unique rate of exchange between its assets, and trades should change that rate gradually rather than abruptly.

▶ **Axiom 1** (Continuity). *The function f is strictly decreasing and (at least) twice-differentiable.*

An AMM must be able to adapt to any market conditions. The *exchange rate* of asset Y in units of X at state $(x, f(x))$ is $-f'(x)$, the negative of the curve's slope at that point.

▶ **Axiom 2** (Expressivity). *The exchange rate $-f'(x)$ can assume every value in the open interval $(0, \infty)$.*

Slippage should work to the disadvantage of the trader. To prevent runaway trading, buying more of asset X should make X more expensive, not less.

▶ **Axiom 3** (Convexity). *For every AMM $A := (x, f(x))$, f is strictly convex.*

It can be shown [12] that for any AMM satisfying these axioms, every valuation has a unique stable state.

▶ **Theorem 1** (Stability). *The function*

$$\phi(v) = f'^{-1}\left(-\frac{v}{1-v}\right) \tag{2}$$

is a homeomorphism $\phi : (0,1) \to \mathbb{R}_{>0}$ that carries each valuation $v \in (0,1)$ to the unique stable state x that minimizes $vx + (1-v)f(x)$.

For example, the stable state map for the constant-product AMM $(x, 1/x)$ is $\phi(v) = \sqrt{\frac{(1-v)}{v}}$. Sometimes it is convenient to express ϕ in vector form as $\Phi : (0,1) \to \mathbb{R}^2_{>0}$, where $\Phi(v) := (\phi(v), f(\phi(v)))$. We will sometimes use $\psi : \mathbb{R}_{>0} \to (0,1)$, the inverse function of ϕ:

$$\psi(x) = -\frac{f'(x)}{1 - f'(x)}. \tag{3}$$

The vector form is $\Psi(x) := (\psi(x), 1 - \psi(x))$.

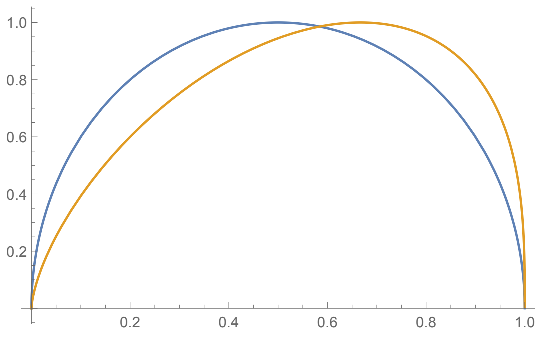


Figure 1 Balanced Capitalization of AMMs $(x, 1/x)$ (symmetric curve, blue) and $(x, 1/x^2)$ (asymmetric curve, yellow) at stable points.

Most of the properties of interest in this paper can be expressed either in the asset domain, as functions of x and $f(x)$, or in the valuation domain, as functions of v and $1-v$.

These definitions extend naturally to AMMs that trade more than two assets. Many (but not all) if the results presented below also generalize, but for brevity we focus on AMMs that trade between two assets.

3 Properties of Interest

3.1 AMM Capitalization

Let $(v, 1-v)$ be a valuation with stable point $(x, f(x))$. What is a useful way to define the *capitalization* (total value) of an AMM's holdings? It may be appealing to pick one asset to act as *numéraire*, computing the AMM's capitalization at point $(x, f(x))$ in terms of that asset alone:

$$\text{numcap}_X(v; A) := \phi(v) + \left(\frac{1-v}{v}\right) f(\phi(v)).$$

For example, the numéraire capitalization of the constant-product AMM at the stable point for valuation v is $2\sqrt{\frac{1}{v}-1}$.

Unfortunately, this notion of capitalization can lead to counter-intuitive results if the numéraire asset becomes volatile. As the value of the numéraire X tends toward zero, “bad money drives out good”, and arbitrage traders will replace more valuable Y with less valuable X. As the AMM fills up with increasingly worthless units of X, its numéraire capitalization grows without bound, so an AMM whose holdings have become worthless has infinite numéraire capitalization.

A more robust approach is to choose a formula that balances the asset classes in proportion to their relative valuations. Define the (balanced) *capitalization* at $(x, f(x))$ to be the sum of the two asset pools weighted by their relative values. Let $\boldsymbol{v} = (v, 1-v)$ and $\boldsymbol{x} = (x, f(x))$.

$$\text{cap}(x, v; A) := vx + (1-v)f(x) = \boldsymbol{v} \cdot \boldsymbol{x}. \tag{4}$$

If v is the current market valuation, then A will usually be in the corresponding stable state $\Phi(v) = (\phi(v), f(\phi(v)))$, yielding

$$\text{cap}(v; A) := \text{cap}(\phi(v), v; A) = \boldsymbol{v} \cdot \Phi(v). \tag{5}$$

For example, AMMs $A := (x, 1/x)$ and $B := (x, 1/x^2)$ have capitalization at their stable points:

$$\begin{aligned}\text{cap}(v; (x, 1/x)) &= 2\sqrt{v(1-v)},\\ \text{cap}(v; (x, 1/x^2)) &= \frac{3v\sqrt[3]{\frac{1}{v}-1}}{2^{2/3}}.\end{aligned}$$

See Figure 1. Both have minimum capitalization 0 at $v = 0$ and $v = 1$, where one asset is worthless, and maximal capitalizations at respective valuations $v = 1/2$ and $v = 2/3$.

▶ **Theorem 2.** *An AMM's capitalization is maximal at the fixed-point $\phi(v) = f(\phi(v))$, where the amounts of X and Y are equal.*

However, Figure 1 shows that the valuation that maximizes an AMM's capitalization is not necessarily $\frac{1}{2}$. An AMM is *symmetric* if $f = f^{-1}$. (Both Uniswap and Curve use symmetric curves.)

▶ **Lemma 3.** *The stable state map for any symmetric AMM satisfies $f(\phi(v)) = \phi(1-v)$*

▶ **Theorem 4.** *Any symmetric AMM has maximum capitalization at $\boldsymbol{v} = (\frac{1}{2}, \frac{1}{2})$.*

3.2 Divergence Loss

Consider the following simple game. A liquidity provider funds an AMM A, leaving it in the stable state $\boldsymbol{x}$ for the current market valuation $\boldsymbol{v}$. Suppose the market valuation changes from $\boldsymbol{v}$ to $\boldsymbol{v}'$, and a trader submits an arbitrage trade that would take A to the stable state $\boldsymbol{x}'$ for the new valuation. The provider has a choice: (1) immediately withdraw its liquidity from A instead of accepting the trade, or (2) accept the trade. It is not hard to see that the provider should always choose to withdraw. The shift to the new valuation changes the AMM's capitalization. If $\boldsymbol{x}$ is the stable state for $\boldsymbol{v}$, then it cannot be stable for $\boldsymbol{v}'$, so the difference between the capitalizations $\boldsymbol{v}' \cdot \boldsymbol{x} - \boldsymbol{v}' \cdot \boldsymbol{x}'$ must be positive, so the arbitrage trader would profit at the provider's expense. (In practice, a provider would take into account the value of current and future fees in making this decision.)

Define the *divergence loss* for an AMM $A = (x, f(x))$ to be

$$\begin{aligned}\text{divloss}(v, v'; A) &:= \boldsymbol{v}' \cdot \Phi(v) - \boldsymbol{v}' \cdot \Phi(v')\\ &= v'\phi(v) + (1-v')f(\phi(v)) - (v'\phi(v') + (1-v')f(\phi(v')))\end{aligned}$$

where $\Phi(v, 1-v) = (\phi(v), f(\phi(v)))$. Sometimes it is useful to express divergence loss in the trade domain, as a function of liquidity pool size instead of valuation:

$$\begin{aligned}\text{divloss}^*(x, x'; A) &:= \text{divloss}(\psi(x), \psi(x'); A)\\ &= \Psi(x') \cdot \boldsymbol{x} - \Psi(x') \cdot \boldsymbol{x}'\\ &= \psi(x')x + (1-\psi(x'))f(x) - (\psi(x')x' + (1-\psi(x'))f(x').\end{aligned}$$

Informally, divergence loss measures the difference in value between funds invested in an AMM and funds left in a wallet. Recall that by definition $\boldsymbol{v}'$ minimizes $\Phi(v') \cdot \boldsymbol{v}'$, so divergence loss is always positive when $\boldsymbol{v} \neq \boldsymbol{v}'$.

Is it possible to bound divergence loss by bounding trade size? More precisely, can an AMM guarantee that any trade that adds δ or fewer units of X incurs a divergence loss less than some ϵ, for positive constants δ, ϵ?

Unfortunately, no. There is a strong sense in which divergence loss can be shifted, but never eliminated. For example, for the constant-product AMM $A := (x, 1/x)$, the divergence loss for a trade of size δ is

$$\text{divloss}^*(x, x+\delta; A) = \frac{\delta^2}{2\delta x^2 + x^3 + \delta^2 x + x} \tag{6}$$

Holding δ constant and letting x approach 0, the divergence loss for even constant-sized trades grows without bound. This property holds for all AMMs.

▶ **Theorem 5.** *No AMM can bound divergence loss even for bounded-size trades.*

Proof. For AMM $A := (x, f(x))$,

$$\begin{aligned}\text{divloss}^*(x, x+\delta; A) = \psi(x+\delta)x + (1 - \psi(x+\delta))f(x) \\ - (\psi(x+\delta)(x+\delta) + (1 - \psi(x+\delta))f(x+\delta)).\end{aligned}$$

Note that $\lim_{x\to 0}(1 - \psi(x+\delta)) > 0$, and $\lim_{x\to 0} f(x) = \infty$. All other terms have finite limits, so $\lim_{x\to 0} \text{divloss}^*(x, x+\delta; A) = \infty$. ◀

What is a provider's worst-case exposure to divergence loss? Consider an AMM $A := (x, f(x))$ in state $(a, f(a))$. As the X asset becomes increasingly worthless, the valuation v' approaches $(0, 1)$ as its stable state approaches $(\infty, 0)$.

$$\lim_{v'\to 0} \text{divloss}(v, v'; A) = (v', 1 - v') \cdot (a, f(a)) - (v', 1 - v') \cdot (\phi(v'), f(\phi(v')))) = f(a).$$

Symmetrically, if the Y asset becomes worthless,

$$\lim_{v'\to 1} \text{divloss}(v, v'; A) = (v', 1 - v') \cdot (a, f(a)) - (v', 1 - v') \cdot (\phi(v'), f(\phi(v')))) = a.$$

The provider's worst-case exposure to divergence loss in state $(a, f(a))$ is thus $\max(a, f(a))$. The *minimum* worst-case exposure occurs when $a = f(a)$. Recall from Section 3.1 that this fixed-point is exactly the state that maximizes the AMM's capitalization. (See Appendix Section C for a more formal treatment of this claim.)

Divergence loss is sometimes called *impermanent loss*, because the loss vanishes if the assets return to their original valuation. The inevitability of impermanent loss does not mean that an AMM's capitalization cannot increase, only that there is always an opportunity cost to the provider for not cashing in earlier.

3.3 Linear Slippage

Linear slippage measures how increasing the size of a trade diminishes that trader's rate of return. Let $A := (x, f(x))$ be an AMM in stable state $(a, f(a))$ for valuation v. Suppose a trader sends $\delta > 0$ units of X to A, taking A from $(a, f(a))$ to $(a+\delta, f(a+\delta))$, the stable state for valuation $v' < v$. If the rate of exchange were linear, the trader would receive $-\delta f'(a)$ units of Y in return for δ units of X. In fact, the trader receives only $f(a) - f(a+\delta)$ units, for a difference of $-\delta f'(a) - f(a) + f(a+\delta)$.

The *linear slippage*(with respect to X) is the value of this difference:

$$\text{linslip}_X(v, v'; A) = \left(\frac{1 - v'}{1 - v}\right)(\boldsymbol{v} \cdot \Phi(v') - \boldsymbol{v} \cdot \Phi(v)) \tag{7}$$

In the trade domain, $\text{linslip}^*_X(x, x'; A) = \text{linslip}_X(\psi(x), \psi(x'); A)$. Linear slippage with respect to Y is defined symmetrically.

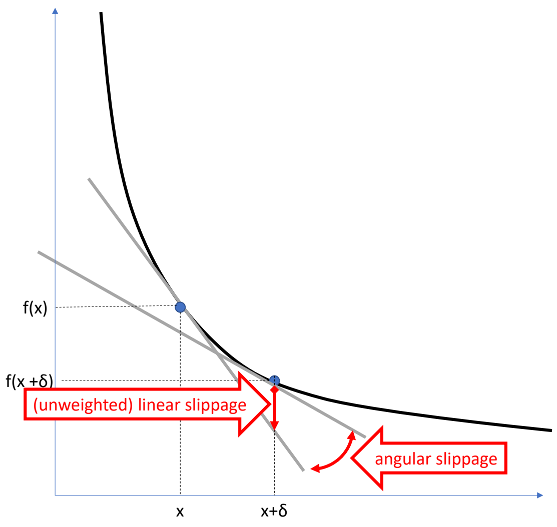


Figure 2 Angular vs linear slippage).

For example, the linear slippage for the constant-product AMM $A := (x, 1/x)$ for a trade of size δ is

$$\text{linslip}^*_X(x, x+\delta; A) = -\frac{\delta^2(\delta + x)}{x^2(\delta^2 + x^2 + 2\delta x + 1)} \tag{8}$$

Just as for divergence loss, holding δ constant and letting x approach 0, the linear slippage across constant-sized trades grows without bound.

▶ **Theorem 6.** *No AMM can bound linear slippage for bounded-size trades.*

Proof. As in the proof of Theorem 5, the claim follows because $\lim_{x->0} \text{linslip}^*_X(x, x+\delta; A) = \infty$. ◀

3.4 Angular Slippage

Angular slippage measures how the size of a trade affects the exchange rate between the two assets. This measure focuses on how a trade affects the traders who come after. Recall that the (instantaneous) exchange rate in state $(x, f(x))$ is the slope of the tangent $f'(x)$. Let $\theta(x)$ denote the angle of that tangent with the X-axis. (We could equally well use the tangent's angle with the Y-axis.) A convenient way to measure the change in price is to measure the change in angle. Consider a trade that carries A from valuation v with stable point $(x, f(x))$, to valuation v' with stable point $(x', f(x'))$. Define the *angular slippage* of that trade to be the difference in tangent angles at x and x' (expressed in the valuation domain):

$$\text{angslip}(v, v'; A) = \theta(\phi(v')) - \theta(\phi(v)).$$

In the trade domain, $\text{angslip}^*(x, x'; A) := \text{angslip}(\phi(x), \phi(x'); A)$. Angular slippage is *additive*: for distinct valuations $v < v' < v''$, $\text{angslip}(v, v''; A) = \text{angslip}(v, v'; A) + \text{angslip}(v', v''; A)$. Note that linear slippage is not additive.

Angular slippage and linear slippage are different ways of measuring the same underlying phenomenon: their relation is illustrated in Figure 2.

Here is how to compute angular slippage. By definition,$\tan\theta(x) = -\frac{1}{f'(x)}$. Let $x' > x$, $\boldsymbol{v} = (v, 1-v)$, and $\boldsymbol{v}' = (v', 1-v')$.

$$\text{angslip}(v, v'; A) = \theta(x') - \theta(x) = \arctan\left(\frac{v - v'}{\boldsymbol{v} \cdot \boldsymbol{v}'}\right) \tag{9}$$

The next lemma says says that the overall angular slippage, $\text{angslip}(0, \infty)$, is a constant independent of the AMM.

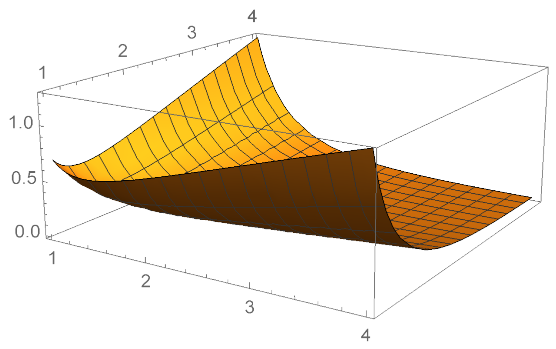


Figure 3 Expected load for $A := (x, 1/x)$ at $1/2$ as a function of β distribution parameters. (Mathematica source is shown in Appendix Section D.)

▶ **Theorem 7.** *For every AMM A,* $\text{angslip}(0, \infty; A) = \pi/2$.

Proof. Consider an AMM $A := (x, f(x))$. As $\lim_{x\to 0} f'(x) = -\frac{v}{1-v} = -\infty$, implying $v = 1$. As $\lim_{x\to\infty} f'(x) = -\frac{v}{1-v} = 0$ implying $v = 0$. Because $\tan \text{angslip}(0, \infty; A) = \frac{1-0}{(0,1)\cdot(0,1)} = \infty$, $\text{angslip}(0, \infty; A) = \frac{\pi}{2}$. ◀

The additive property means that no AMM can eliminate angular slippage over every finite interval. Lowering angular slippage in one interval requires increasing it elsewhere.

▶ **Corollary 8.** *For any AMM A, and any level of slippage s, $0 < s < \pi/2$, there is an interval $(x_0, x_1) \subset \mathbb{R}_{>0}$ such that* $\text{angslip}^*(x_0, x_1; A) > s$.

For example, the Curve [11] AMM advertises itself as having lower slippage than its competitors. Theorem 7 helps us understand this claim: compared to a constant-product AMM, Curve does have lower slippage than a constant-product AMM for stable coins when they trade at near-parity, but it must have higher slippage when the exchange rate wanders out of that interval.

3.5 Load

Divergence loss is a cost to providers, and linear slippage is a cost to traders. Controlling one without controlling the other is pointless because AMMs function only if both providers and traders consider their costs acceptable. We propose the following measure to balance provider-facing and trader-facing costs. The *load* (with respect to X) across an interval is the product of that interval's divergence loss and linear slippage:

$$\text{load}_X(v, v'; A) := \text{divloss}(v, v'; A)\, \text{linslip}_X(v, v'; A) \tag{10}$$

Load can also be expressed in the trade domain:
$\text{load}^*_X(x, x'; A) := \text{divloss}^*(x, x'; A)\, \text{linslip}^*_X(x, x'; A)$.

3.6 Expected Load

We have seen that cost measures such as divergence loss, linear slippage, angular slippage, and load cannot be bounded in the worst case. Nevertheless, these costs can be shifted. Not all AMM states are equally likely. For example, one would expect stablecoins to trade at near parity [11].

Suppose we are given a probability density for future valuations. This distribution might be given *a priori*, or it may be learned from historical data. Can we compare the behavior of alternative AMMs given such a distribution?

Let $p(v)$ be the distribution over possible future valuations. The expected load when trading X for Y starting in the stable state for valuation $\boldsymbol{v}$ is

$$\int_0^v P(v'|v' < v)\,\mathrm{load}_X(v, v'; A)dv'$$

Weighting this expectation with the probability $P(v > v')$ that the trade will go in that direction yields

$$P(v > v') \int_0^v P(v'|v' < v)\,\mathrm{load}_X(v, v'; A)dv' = \int_0^v p(v')\,\mathrm{load}_X(v, v'; A)dv'.$$

Define the *expected load* of AMM A at valuation v to be:

$$E_p[\mathrm{load}(v; A)] := \int_0^v p(v')\,\mathrm{load}_X(v, v'; A)dv' + \int_v^1 p(v')\,\mathrm{load}_Y(v, v'; A)dv'.$$

Of course, one can compute the expected value of any the measures proposed here, not just load.

Figure 3 shows the expected load for $A := (1, 1/x)$, starting at valuation $1/2$, where the expectation is taken over the beta(α_1, α_2) distributions [22], where parameters α_1, α_2 range independently from 1 to 4. Inspecting the figure shows that symmetric distributions $\beta(\alpha, \alpha)$, which are increasingly concentrated around $1/2$ as α grows, yield decreasing loads as the next valuation becomes increasingly likely to be close to the current one. By contrast, asymmetric distributions, which favor unbalanced valuations, yield higher loads because the next valuation is likely to be farther from the current one.

4 Sequential Composition

The *sequential composition* of two AMMs is constructed by using the output of one AMM as the input to the other. (See Engel and Herlihy [12] for a proof that the sequential composition of two AMMs is an AMM.) For example, if A trades between florins and guilders, and B trades between guilders and francs, then their sequential composition $A \oplus B$ trades between florins and francs. A trader might deposit florins in A, receiving guilders, then deposit those guilders in B, receiving francs. In this section, we investigate how linear slippage and divergence loss interact with sequential composition.

Consider two AMMs $A = (x, f(x)), B = (y, g(y))$, where A trades between X and Y, and B between Y and Z. If A is in state $(a, f(a))$ and B in state $(b, g(b))$ then their sequential composition is $A \oplus B := (x, h(x))$, where $h(x) = g(b + f(a) - f(x))$ [12]. (The sequential composition of more than two AMMs can be constructed by repeated two-way compositions.)

Let $\boldsymbol{v} = (v_1, v_2, v_3)$ be the market valuation linking X, Y, Z, inducing pairwise valuations

$$v_{12} = \frac{v_1}{v_1 + v_2},\ v_{23} = \frac{v_2}{v_2 + v_3},\ v_{13} = \frac{v_1}{v_1 + v_3},$$

along with their vector forms $\boldsymbol{v}_{12} = (v_{12}, 1 - v_{12})$, $\boldsymbol{v}_{23} = (v_{23}, 1 - v_{23})$, $\boldsymbol{v}_{13} = (v_{13}, 1 - v_{13})$. Let $\boldsymbol{v}' \neq \boldsymbol{v}$ be a three-way valuation inducing analogous pair-wise valuations. Let $\phi_A, \phi_B, \phi_{AB} : (0, 1) \to \mathbb{R}_{>0}$ be the stable point maps for $A, B, A \oplus B$ respectively, and $\Phi_A, \Phi_B, \Phi_{AB} : (0, 1) \to \mathbb{R}^2_{>0}$ their vector forms.

Our composition rules apply when A and B start in their respective stable states[1] for a market valuation $\boldsymbol{v}$: $(a, f(a))$ is the stable state for v_{12}, $(a+\delta, f(a+\delta))$ for v'_{12}. $(b, g(b))$ for v_{23}, and $(b + f(a) - f(a+\delta), g(b + f(a) - f(a+\delta))$ for v'_{23}. We analyze the changes in divergence loss and linear slippage when the market valuation changes from $\boldsymbol{v}$ to $\boldsymbol{v}'$.

4.1 Divergence Loss

Initially, the combined capitalization of A and B is $v_1 x + v_2(f(x) + y) + v_3 g(y)$. A trader sends δ units to A, reducing the combined capitalization by

$$-\delta v'_1 + v'_2(f(x) - f(x+\delta)) = v'_3 \operatorname{divloss}(v_{12}, v'_{12}; A) \tag{11}$$

Next the trader sends the assets returned from the first trade to B, reducing the combined capitalization by:

$$v'_2(f(x+\delta) - f(x)) + v'_3(g(y) - g(y + f(x) - f(x+\delta))) = v'_1 \operatorname{divloss}(v_{23}, v'_{23}; B) \tag{12}$$

Finally, treating both trades as a single transaction reduces the combined capitalization by:

$$-\delta v'_1 + v'_3(h(x) - h(x+\delta)) = v'_2 \operatorname{divloss}(v_{13}, v'_{13}; A \oplus B) \tag{13}$$

Combining Equations 11-13 yields

$$\operatorname{divloss}(v_{13}, v'_{13}; A \oplus B) = \left(\frac{v'_3}{v'_2}\right) \operatorname{divloss}(v_{12}, v'_{12}; A) + \left(\frac{v'_1}{v'_2}\right) \operatorname{divloss}(v_{23}, v'_{23}; B). \tag{14}$$

The effect of sequential composition on divergence loss is linear but not additive: the divergence loss of the composition is a weighted sum of the divergence losses of the components.

4.2 Linear Slippage

With respect to $\boldsymbol{v}$, a trader who sends δ units of X to A incurs the following slippage

$$v'_2(-\delta f'(x) + f(x+\delta) - f(x)) = (v'_1 + v'_2) \operatorname{linslip}_X(v'_{12}, v_{12}; A). \tag{15}$$

Next the trader sends the assets returned from the first trade to B, incurring the following slippage:

$$v'_3\left((f(x) - (x+\delta))g'(y) + g(y + f(x) - (x+\delta)) - g(y)\right) = (v'_2 + v'_3) \operatorname{linslip}_X(v_{23}, v'_{23}; B). \tag{16}$$

Finally, treating both trades as a single transaction yields slippage:

$$v'_3(-\delta h'(x) + h(x+\delta) - h(x))(v'_1 + v'_3) \operatorname{linslip}_X(v'_{13}, v_{13}, A \oplus B) \tag{17}$$

Combining Equations 15-17 yields

$$\operatorname{linslip}_X(v_{13}, v'_{13}; A \oplus B) = \left(\frac{1 - v'_3}{1 - v'_2}\right) \operatorname{linslip}_X(v_{12}, v'_{12}; A) + \left(\frac{1 - v'_1}{1 - v'_2}\right) \operatorname{linslip}_X(v_{23}, v'_{23}; B) \tag{18}$$

[1] If A and B do not start in stable states for the current market valuation, then an arbitrage trader will eventually put them there.

4.3 Angular Slippage

A trader sends δ to A, where $f(a+\delta)$ is the stable point for v'_{12}, $g(b+f(a)-f(a+\delta))$ the stable point for v'_{23}. By construction,

$$h'(a) = -g'(b)f'(a) = -\frac{v_{23}}{1-v_{23}}\frac{v_{12}}{1-v_{12}} = -\frac{v_2}{v_3}\frac{v_1}{v_2} = -\frac{v_1}{v_3}$$
$$h'(a+\delta) = -g'(b+f(a)-f(a+\delta))f'(a+\delta) = -\frac{v'_{23}}{1-v'_{23}}\frac{v'_{12}}{1-v'_{12}} = -\frac{v'_2}{v'_3}\frac{v'_1}{v'_2} = -\frac{v'_1}{v'_3}$$

Define $\theta_A(x), \theta_B(y), \theta_B(y)$ to be the respective angles of $f'(x)$, $g'(y)$, and $h'(x)$ with their X-axes. We can express the tangents of the composite AMM's angles in terms of the tangents of the component AMMs' angles.

$$\tan\theta_{AB}(x) = -\frac{1}{h'(x)} = -\frac{1}{-g'(b+f(a)-f(x))f'(x)}$$
$$\tan\theta_{AB}(a) = \frac{v_3}{v_2}\frac{v_2}{v_1} = \frac{v_3}{v_1}, \qquad \tan\theta_{AB}(a+\delta) = \frac{v'_3}{v'_2}\frac{v'_2}{v'_1} = \frac{v'_3}{v'_1}.$$

The component AMMs A and B determine the valuations v_1, v_2, v'_1, v'_2, which induce the remaining valuations $v_3, v'_3, v_{12}, v_{23}, v_{13}, v'_{12}, v'_{23}, v'_{13}$.

$$\text{angslip}(v_{13}, v_{13'}, A \oplus B) = \arctan\left(\frac{v_{13}-v'_{13}}{\boldsymbol{v}_{13}\cdot\boldsymbol{v}'_{13}}\right) \tag{19}$$

It follows that the angular slippage of the sequential composition of two AMMs can be computed from the component AMMs' valuations.

4.4 Load

Combining Equation 14 and Equation 18 yields

$$\text{load}_X(v_{13}, v'_{13}; A\oplus B) = \left(\frac{v'_3(1-v'_3)}{v'_2(1-v'_2)}\right)\text{load}_X(v_{12}, v'_{12}; A) + \left(\frac{v'_1(1-v'_1)}{v'_2(1-v'_2)}\right)\text{load}_X(v_{23}, v'_{23}; B) \tag{20}$$
$$+\left(\frac{v'_3(1-v'_3)}{v'_2(1-v'_2)}\right)\text{divloss}(v_{12}, v'_{12}; A)\,\text{linslip}_X(v_{23}, v'_{23}; B)$$
$$+\left(\frac{v'_1(1-v'_1)}{v'_2(1-v'_2)}\right)\text{divloss}(v_{23}, v'_{23}; B)\,\text{linslip}_X(v_{12}, v'_{12}; A)$$

It follows that the load of a sequential composition is a weighted sum of the loads of the components, plus additional (strictly positive) cross-terms.

5 Parallel Composition

Parallel composition [12] arises when a trader is presented with two AMMs $A := (x, f(x))$ and $B := (x, g(x))$, both trading assets X and Y, and seeks to treat them as a single combined AMM $A||B$. Let $\phi_A, \phi_B : (0,1) \to \mathbb{R}_{>0}$ be the stable point maps for A, B respectively, with $\Phi_A, \phi_B : (0,1) \to \mathbb{R}^2_{>0}$ their vector forms, and $\psi_A, \psi_B : \mathbb{R}_{>0} \to (0,1)$ their inverses. (The parallel composition of more than two AMMs can be constructed by repeated two-way compositions.)

As shown elsewhere [12], a trader who sends δ units of X to the combined AMM maximizes return by splitting those units between A and B, sending $t\delta$ to A and $(1-t)\delta$ to B, for $0 \le t \le 1$, where $f'(x+t\delta) = g'(y+(1-t)\delta)$.

We assume traders are rational, and always split trades in this way. Because the derivatives are equal, $x + t\delta$ and $y + (1 - t)\delta$ are stable points of A and B respectively for the same valuation v', so $x + t\delta = \phi_A(v')$, and $y + (1 - t)\delta = \phi_B(v')$. If A is in state $(a, f(a))$ and B in $(b, g(b))$, then $A||B := (x, h(x))$, where $h(x) := (f(a) - f(a + tx) + g(b) - g(b + (1 - t)x)$. Our composition rules apply when both A, B are in their stable states for valuation $\boldsymbol{v} = (v, 1 - v)$. We analyze the change in divergence loss and linear slippage when the common valuation changes from $\boldsymbol{v}$ to $\boldsymbol{v}' = (v', 1 - v')$. The new valuation $\boldsymbol{v}'$ may be the new market valuation, or it may be the best the trader can reach with a fixed budget of δ.

5.1 Divergence Loss

If a trader sends δ units of X to $A||B$, the combined capitalization suffers a loss of

$$\text{divloss}(v, v'; A||B) = \text{divloss}(v, v'; A) + \text{divloss}(v, v'; B). \tag{21}$$

It follows that divergence loss under parallel composition is additive.

5.2 Linear Slippage

A straightforward calculation shows:

$$\text{linslip}_X(v, v'; A||B) = \text{linslip}_X(v, v'; A) + \text{linslip}_X(v, v'; B) \tag{22}$$

Linear slippage is thus additive under parallel composition.

Linear slippage is also linear under scalar multiplication. Any AMM $A := (x, f(x))$ can be *scaled* by a constant $\alpha > 0$ yielding a distinct AMM $\alpha A := (\alpha x, \alpha f(x))$. Let $(x, f(x))$ be the stable point for valuation $\boldsymbol{v}$, and $(x', f'(x'))$ the stable point for $\boldsymbol{v}'$.

$$\text{linslip}_X(v, v'; \alpha A) = \alpha\,\text{linslip}_X(v, v'; A). \tag{23}$$

5.3 Angular Slippage

Because both A and B go from stable states for v to stable states for v',

$$f'(a) = g'(a) = h'(a) = \frac{-v}{1 - v}, \quad f'(a + t\delta) = g'(a + (1 - t)\delta) = h'(a + \delta) = \frac{-v'}{1 - v'}$$

It follows that

$$\text{angslip}(v, v'; A||B) = \text{angslip}(v, v'; A) = \text{angslip}(v, v'; B). \tag{24}$$

5.4 Load

Combining Equation 21 and Equation 22 yields

$$\begin{aligned}\text{load}_X(v, v'; A||B) &= \text{load}_X(v, v'; A) + \text{load}_X(v, v'; B) \\ &\quad \text{divloss}(v, v'; A)\,\text{linslip}_X(v, v'; B)\,\text{divloss}(v, v'; B)\,\text{linslip}_X(v, v'; A)\end{aligned} \tag{25}$$

It follows that the parallel composition's load is the sum of the components' loads, plus additional (strictly positive) cross-terms.

6 Adaptive Strategies

So far we have proposed several ways to quantify the costs associated with AMMs. Now we turn our attention to strategies for adapting to cost changes. A complete analysis of adaptive AMM strategies is material for another paper, so here we summarize two broad strategies motivated by our proposed cost measures. We focus on adjustments that might be executed automatically, without demanding additional liquidity from providers.

6.1 Change of Valuation

Suppose an AMM learns, perhaps from a trusted Oracle service, that its assets' market valuation has moved away from the AMM's current stable state, leaving the providers exposed to substantial divergence loss. Specifically, suppose $A := (x, f(x))$ has valuation v_1 with stable state (a_1, b_1), when it learns that the market valuation has changed to v_2 with stable point (a_2, b_2).

An arbitrage trader would move A from (a_1, b_1) to (a_2, b_2), pocketing a profit. Informally, A can eliminate that divergence loss by "pretending" to conduct that arbitrage trade itself, leaving the state the same, but moving the curve. We call this strategy *pseudo-arbitrage*.

A changes its function using linear changes of variable in x and y. Suppose $a_1 > a_2$ and $b_2 > b_1$. First, replace x with $x - (a_1 - a_2)$, shifting the curve along the X-axis. Next, replace y with $y - (b_2 - b_1)$, shifting the curve along the y-axis. The transformed AMM is now $A' := (x, f(x - (a_1 - a_2)) - (b_2 - b_1))$. The current state (a_1, b_1) still lies on the shifted curve, but now with slope $\frac{v_2}{v_2 - 1}$, matching the new valuation. The advantage of this change is that A's providers are no longer exposed to divergence loss from the new market valuation.

The disadvantage is that pseudo-arbitrage produces AMMs that do not satisfy the usual boundary conditions $f(0) = \infty$ and $f(\infty) = 0$, although they continue to satisfy the AMM axioms. In practical terms, A now has more units of X than it needs, but not enough units of Y to cover all possible trades. The AMM must refuse trades that would lower its Y holdings below zero, and there are $(a_1 - a_2)$ units of X inaccessible to the AMM. The liquidity providers might withdraw this excess, they might "top up" with more units of Y to rebalance the pools, or they might leave the extra balance to cover future pseudo-arbitrage changes. (Note that A's ability to conduct trades only while the valuation stays within a certain range is similar to Uniswap v3's "concentrated liquidity" option.)

6.2 Change of Distribution

Suppose an AMM's formula was initially chosen to match a predicted distribution on future valuations. If that prediction changes, then it may be possible to adjust the AMM's formula to match the new prediction. Such an adjustment might be built into the AMM's smart contract, or it could be imposed from outside by the liquidity providers. The AMM's current function could be replaced with an alternative that improves some expected cost measure, say, reducing expected load or increasing expected capitalization. But replacing AMM $A := (x, f(x))$, in the stable state for the market valuation, with another $\widetilde{A} := (x, \widetilde{f}(x))$, must follow certain common-sense rules.

First, any such replacement should not change the AMM's reserves: if the AMM is in state $(a, f(a))$, then the updated AMM is in state $(a, \widetilde{f}(a))$ where $f(a) = \widetilde{f}(a)$. Adding or removing liquidity requires the active participation of the AMM's providers, which can certainly happen, but not as part of the kind of automatic strategy considered here.

Second, any such replacement should not change the AMM's current exchange rate: if the AMM is in state $(a, f(a))$, then the updated AMM is in state $(a, \widetilde{f}(a))$ where $f'(a) = \widetilde{f}'(a)$. To do otherwise invites further divergence loss. If $(a, f(a))$ is the stable state for the current valuation, and $f'(a) \neq \widetilde{f}'(a)$, then $(a, \widetilde{f}(a))$ is not stable, and a trader can make an arbitrage profit (and divergence loss) by moving the AMM's state back to the stable state.

For example, an AMM's expected capitalization under distribution p is

$$\int_0^1 p(v)\boldsymbol{v} \cdot \Phi(v)dv,$$

where $\Phi(v) = (\phi(v), f(\phi(v)))$. If the distribution changes to $\widetilde{p}$, then an adaptive strategy is to find a function $\widetilde{f} : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ with associated stable-point function $\widetilde{\phi} : (0,1) \to \mathbb{R}_{>0}$ that optimizes (or at least improves) the difference

$$\int_0^1 p(v)\boldsymbol{v} \cdot \Phi(v)dv - \int_0^1 \widetilde{p}(v)\boldsymbol{v} \cdot \widetilde{\Phi}(v)dv,$$

subject to boundary conditions $f(a) = \widetilde{f}(a)$ and $f'(a) = \widetilde{f}'(a)$, where $\widetilde{\Phi}(v) = (\widetilde{\phi}(v), f(\widetilde{\phi}(v)))$. Developing practical ways to find such functions is the subject of future work.

7 Related Work

Today, the most popular automated market maker is *Uniswap* [2, 5, 16, 23], a family of constant-product AMMs. Originally trading between ERC-20 tokens and ether cryptocurrency, later versions added direct trading between pairs of ERC-20 tokens, and allowed liquidity providers to restrict the range of prices in which their asset participate. *Bancor* [17] AMMs permit more flexible pricing schemes, and later versions [7] include integration with external "price oracles" to keep prices in line with market conditions. *Balancer* [18] AMMs trade across more than two assets, based on a *constant mean* formula that generalizes constant product. *Curve* [11] uses a custom curve specialized for trading *stablecoins* , maintaining low slippage and divergence loss as long as the stablecoins trade at near-parity. Pourpouneh *et al.* [21] provide a survey of current AMMs.

The formal model for AMMs used here, including the axioms constraining AMM functions, and notions of composition, are taken from Engel and Herlihy [12].

Angeris and Chitra [3] introduce a *constant function market maker* model and focus on conditions that ensure that agents who interact with AMMs correctly report asset prices.

In *event prediction markets* [1, 9, 10, 14, 15], parties effective place bets on the outcomes of certain events, such as elections. Event prediction AMMs differ from DeFi AMMs in important ways: pricing models are different because prediction outcome spaces are discrete rather than continuous, prediction securities have finite lifetimes, and composing AMMs is not a concern.

AMM curves resemble *consumer utility curves* from classical economics [19], and trader arbitrage resembles *expenditure minimization*. Despite some mathematical similarities, there are fundamentally differences in application. In particular, traders interact with AMMs via composition, an issue that does not arise in the consumer model.

Aoyagi [6] analyzes strategies for constant-product AMM liquidity providers in the presence of "noise" trading, which is not intended to move prices, and "informed" trading, intended to move the AMM to the stable point for a new and more accurate valuation.

Angeris *et al.* [4] propose an economic model relating how the curvature of the AMM's function affects LP profitability in the presence of informed and uninformed traders.

Bartoletti *et al.* [8] give a formal semantics for a constant-product AMM expressed as a labeled transition system, and formally verify a number of basic properties.

References

1 Jacob Abernethy, Yiling Chen, and Jennifer Wortman Vaughan. An optimization-based framework for automated market-making. In *Proceedings of the 12th ACM conference on Electronic commerce - EC '11*, page 297, San Jose, California, USA, 2011. ACM Press. `doi:10.1145/1993574.1993621`.

2 Hayden Adams, Noah Zinsmeister, and Dan Robinson. Uniswap v2 core. `https://uniswap.org/whitepaper.pdf`, March 2020. As of 8 February 2021.

3 Guillermo Angeris and Tarun Chitra. Improved Price Oracles: Constant Function Market Makers. *SSRN Electronic Journal*, 2020. `doi:10.2139/ssrn.3636514`.

4 Guillermo Angeris, Alex Evans, and Tarun Chitra. When does the tail wag the dog? Curvature and market making. *arXiv:2012.08040 [q-fin]*, December 2020. `arXiv:2012.08040`.

5 Guillermo Angeris, Hsien-Tang Kao, Rei Chiang, Charlie Noyes, and Tarun Chitra. An analysis of Uniswap markets. *arXiv:1911.03380 [cs, math, q-fin]*, February 2021. `arXiv:1911.03380`.

6 Jun Aoyagi. Lazy Liquidity in Automated Market Making. *SSRN Electronic Journal*, 2020. `doi:10.2139/ssrn.3674178`.

7 Bancor. Proposing bancor v2.1: Single-sided amm with elastic bnt supply. `https://blog.bancor.network/proposing-bancor-v2-1-single-sided-amm-with-elastic-bnt-supply-bcac9fe655b`, October 2020. As of 8 February 2021.

8 Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch-Lafuente. A theory of Automated Market Makers in DeFi. *arXiv:2102.11350 [cs]*, April 2021. `arXiv:2102.11350`.

9 Yiling Chen and David M. Pennock. A utility framework for bounded-loss market makers. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, UAI'07, pages 49–56, Arlington, Virginia, USA, 2007. AUAI Press.

10 Yiling Chen and Jennifer Wortman Vaughan. A new understanding of prediction markets via no-regret learning. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, EC '10, pages 189–198, New York, NY, USA, 2010. Association for Computing Machinery. `doi:10.1145/1807342.1807372`.

11 Michael Egorov. Stableswap - efficient mechanism for stablecoin liquidity. `https://www.curve.fi/stableswap-paper.pdf`, November 2019. As of 8 February 2021.

12 Daniel Engel and Maurice Herlihy. Composing Networks of Automated Market Makers. *arXiv:2106.00083 [cs]*, June 2021. `arXiv:2106.00083`.

13 Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger, July 2021. URL: `https://ethereum.github.io/yellowpaper/paper.pdf`.

14 Robin Hanson. Combinatorial Information Market Design. *Information Systems Frontiers*, 5(1):107–119, January 2003. `doi:10.1023/A:1022058209073`.

15 Robin Hanson. Logarithmic market scoring rules for modular combinatorial information aggregation. *Journal of Prediction Markets*, 1(1):3–15, 2007. URL: `https://EconPapers.repec.org/RePEc:buc:jpredm:v:1:y:2007:i:1:p:3-15`.

16 Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. Uniswap v3 Core, March 2021. URL: `https://uniswap.org/whitepaper-v3.pdf`.

17 Eyal Hertzog, Guy Benartzi, and Galia Benartzi. Bancor protocol. `https://whitepaper.io/document/52/bancor-whitepaper`, 2017.

18 Fernando Martinelli and Nikolai Mushegian. Balancer: A non-custodial portfolio man- ager, liquidity provider, and price sensor. https://balancer.finance/whitepaper/, 2109. As of 2 February 2021.

19 Andreu Mas-Collell, Michael Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

20 Pintail. Uniswap: A good deal for liquidity providers? https://medium.com/@pintail/uniswap-a-good-deal-for-liquidity-providers-104c0b6816f2.

21 Mohsen Pourpouneh, Kurt Nielsen, and Omri Ross. Automated Market Makers. IFRO Working Paper 2020/08, University of Copenhagen, Department of Food and Resource Economics, July 2020. URL: https://ideas.repec.org/p/foi/wpaper/2020_08.html.

22 wikipedia. Beta distribution. https://en.wikipedia.org/wiki/Beta_distribution. As of 11 August 2021.

23 Yi Zhang, Xiaohong Chen, and Daejun Park. Formal specification of constant product (x . y = k) market maker model and implementation. https://github.com/runtimeverification/verified-smart-contracts/blob/uniswap/uniswap/x-y-k.pdf, 2018.

A Appendix: Derivations of Equations

A.1 Equation 7

$$\begin{aligned}
\mathrm{linslip}_X(v, v'; A) &:= (1-v')\left(-\delta f'(x) + f(x+\delta) - f(x)\right) \\
&= (1-v')\left(\delta \frac{v}{1-v} + f(x+\delta) - f(x)\right) \\
&= \left(\frac{1-v'}{1-v}\right)\left(v\phi(v') - v\phi(v) + (1-v)f(\phi(v')) - (1-v)f(\phi(v))\right) \\
&= \left(\frac{1-v'}{1-v}\right)\left(\boldsymbol{v}\cdot\Phi(v') - \boldsymbol{v}\cdot\Phi(v)\right) \\
&= \left(\frac{1-v'}{1-v}\right)\mathrm{divloss}(v', v; A).
\end{aligned}$$

A.2 Equation 9

$$\begin{aligned}
\mathrm{angslip}(v, v'; A) &= \theta(x') - \theta(x) \\
&= \arctan\left(\frac{1}{-f'(x')}\right) - \arctan\left(\frac{1}{-f'(x)}\right) \\
&= \arctan\left(\frac{1-v'}{v'}\right) - \arctan\left(\frac{1-v}{v}\right) \\
&= \arctan\left(\frac{(\frac{1-v'}{v'}) - (\frac{1-v}{v})}{1 + (\frac{1-v'}{v'})(\frac{1-v}{v})}\right) \\
&= \arctan\left(\frac{v-v'}{\boldsymbol{v}\cdot\boldsymbol{v}'}\right)
\end{aligned}$$

A.3 Equation 11

$$\begin{aligned}
-\delta v_1' + v_2'(f(x) - f(x+\delta)) &= v_1'(\phi_A(v_{12}) - \phi_A(v_{12}')) + v_2'(f(\phi_A(v_{12})) - f(\phi_A(v_{12}'))) \\
&= (v_1' + v_2')(\boldsymbol{v}_{12}' \cdot \Phi_A(v_{12}) - \boldsymbol{v}_{12}' \cdot \Phi_A(v_{12}')) \\
&= (v_1' + v_2')\,\mathrm{divloss}(v_{12}, v_{12}'; A) \\
&= v_3'\,\mathrm{divloss}(v_{12}, v_{12}'; A)
\end{aligned}$$

A.4 Equation 12

$$\begin{aligned}
&v_2'(f(x+\delta)-f(x))+v_3'(g(y)-g(y+f(x)-f(x+\delta)))\\
&\qquad =v_2'(\phi_B(v_{23})-\phi_B(v_{23}'))+v_3'(g(\phi_B(v_{23}))-g(\phi_A(v_{23}'))\\
&\qquad =(v_2'+v_3')(\boldsymbol{v}_{23}'\cdot\Phi_B(v_{23})-\boldsymbol{v}_{23}'\cdot\Phi_B(v_{23}'))\\
&\qquad =v_1'\,\mathrm{divloss}(v_{23},v_{23}';B
\end{aligned}$$

A.5 Equation 13

$$\begin{aligned}
-\delta v_1'+v_3'(h(x)-h(x+\delta)) &= v_1'(\phi_{AB}(v_{13})-\phi_{AB}(v_{13}'))+v_3'(h(\phi_{AB}(v_{13}))-h(\phi_{AB}(v_{13}'))\\
&=(v_1'+v_3')((\boldsymbol{v}_{13}'\cdot\Phi_{AB}(v_{13})-\boldsymbol{v}_{13}'\cdot\Phi_A(v_{13}'))\\
&=v_2'\,\mathrm{divloss}(v_{13},v_{13}';A\oplus B)
\end{aligned}$$

A.6 Equation 14

$$\begin{aligned}
v_2'\,\mathrm{divloss}(v_{13},v_{13}';A\oplus B) &= v_3'\,\mathrm{divloss}(v_{12},v_{12}';A)+v_1'\,\mathrm{divloss}(v_{23},v_{23}';B)\\
\mathrm{divloss}(v_{13},v_{13}';A\oplus B) &= \left(\frac{v_3'}{v_2'}\right)\mathrm{divloss}(v_{12},v_{12}';A)+\left(\frac{v_1'}{v_2'}\right)\mathrm{divloss}(v_{23},v_{23}';B).
\end{aligned}$$

A.7 Equation 15

$$\begin{aligned}
\mathrm{linslip}_X(v_{12},v_{12}';A) &= \left(\frac{v_2'}{v_2}\right)\left(\frac{v_1+v_2}{v_1'+v_2'}\right)\left(\boldsymbol{v}_{12}\cdot\Phi(v_{12}')-\boldsymbol{v}_{12}\cdot\Phi(v_{12})\right)\\
v_2'(-\delta f'(x)+f(x+\delta)-f(x)) &= v_2'\left(\frac{v_1}{v_2}(\phi_A(v_{12}')-\phi_A(v_{12}))+(f(\phi_A(v_{12}'))-f(\phi_A(v_{12}))\right)\\
&=\frac{v_2'}{v_2}(v_1+v_2)(\boldsymbol{v}_{12}\cdot\Phi_A(v_{12}')-\boldsymbol{v}_{12}\cdot\Phi_A(v_{12}))\\
&=(v_1'+v_2')\,\mathrm{linslip}_X(v_{12}',v_{12};A).
\end{aligned}$$

A.8 Equation 16

$$\begin{aligned}
&\mathrm{linslip}_X(v_{23},v_{23}';B)=\left(\frac{v_3'}{v_3}\right)\left(\frac{v_2+v_3}{v_2'+v_3'}\right)(\boldsymbol{v}_{23}\cdot\Phi(v_{23}')-\boldsymbol{v}_{23}\cdot\Phi(v_{23}))\\
&\qquad\qquad v_3'\left((f(x)-(x+\delta))g'(y)+g(y+f(x)-(x+\delta))-g(y)\right)\\
&\qquad\qquad =v_3'\left((f(x)-(x+\delta))\frac{v_2}{v_3}+g(y+f(x)-(x+\delta))-g(y)\right)\\
&\qquad\qquad =\frac{v_3'}{v_3}(v_2+v_3)\left(\boldsymbol{v}_{23}\cdot\Phi_B(v_{23}')-\boldsymbol{v}_{23}\cdot\Phi_B(v_{23})\right))\\
&\qquad\qquad =(v_2'+v_3')\,\mathrm{linslip}_X(v_{23},v_{23}';B).
\end{aligned}$$

A.9 Equation 17

$$\begin{aligned}
v_3'(-\delta h'(x)+h(x+\delta)-h(x)) &= v_3'\left(\frac{v_1}{v_3}(\phi_A(v_{13}')-\phi_A(v_{13}))+(h(\phi_A(v_{13}'))-h(\phi_A(v_{13}))\right)\\
&=\frac{v_3'}{v_3}(v_1+v_3)(\boldsymbol{v}_{13}\cdot\Phi_A(v_{13}')-\boldsymbol{v}_{13}\cdot\Phi_A(v_{13}))\\
&=(v_1'+v_3')\,\mathrm{linslip}_X(v_{13}',v_{13},A\oplus B)
\end{aligned}$$

A.10 Equation 18

$$\begin{aligned}
(v_1' + v_3')\,\mathrm{linslip}_X(v_{13}, v_{13}'; A \oplus B)) &= (v_1' + v_2')\,\mathrm{linslip}_X(v_{12}, v_{12}'; A) \\
&\quad + (v_2' + v_3')\,\mathrm{linslip}_X(v_{13}, v_{13}'; B) \\
\mathrm{linslip}_X(v_{13}, v_{13}'; A \oplus B) &= \left(\frac{v_1' + v_2'}{v_1' + v_3'}\right)\mathrm{linslip}_X(v_{12}, v_{12}'; A) \\
&\quad + \left(\frac{v_2' + v_3'}{v_1' + v_3'}\right)\mathrm{linslip}_X(v_{23}, v_{23}'; B) \\
&= \left(\frac{1 - v_3'}{1 - v_2'}\right)\mathrm{linslip}_X(v_{12}, v_{12}'; A) \\
&\quad + \left(\frac{1 - v_1'}{1 - v_2'}\right)\mathrm{linslip}_X(v_{23}, v_{23}'; B)
\end{aligned}$$

A.11 Equation 18

$$\begin{aligned}
\mathrm{angslip}(v_{13}, v_{13'}, A \oplus B) &= \arctan\left(\frac{v_3'}{v_1'}\right) - \arctan\left(\frac{v_3}{v_1}\right) \\
&= \arctan\left(\frac{v_1 v_3' - v_1' v_3}{v_1 v_1' + v_3 v_3'}\right) \\
&= \arctan\left(\frac{v_{13} - v_{13}'}{\boldsymbol{v}_{13} \cdot \boldsymbol{v}_{13}}\right)
\end{aligned}$$

A.12 Equation 20

$$\begin{aligned}
\mathrm{load}_X(v_{13}, v_{13}'; A \oplus B) &= \mathrm{divloss}(v_{13}, v_{13}'; A \oplus B)\,\mathrm{linslip}_X(v_{13}, v_{13}'; A \oplus B) \\
&= \left(\frac{v_3'(1 - v_3')}{v_2'(1 - v_2')}\right)\mathrm{load}_X(v_{12}, v_{12}'; A) \\
&\quad + \left(\frac{v_1'(1 - v_1')}{v_2'(1 - v_2')}\right)\mathrm{load}_X(v_{23}, v_{23}'; B) \\
&\quad + \left(\frac{v_3'(1 - v_3')}{v_2'(1 - v_2')}\right)\mathrm{divloss}(v_{12}, v_{12}'; A)\,\mathrm{linslip}_X(v_{23}, v_{23}'; B) \\
&\quad + \left(\frac{v_1'(1 - v_1')}{v_2'(1 - v_2')}\right)\mathrm{divloss}(v_{23}, v_{23}'; B)\,\mathrm{linslip}_X(v_{12}, v_{12}'; A)
\end{aligned}$$

A.13 Equation 21

$$\begin{aligned}
\mathrm{divloss}(v, v'; A||B) &= v'(-\delta) + (1 - v')(f(x) - f(x + t\delta) + g(y) - g(y + (1 - t)\delta)) \\
&= -v'(x + t\delta - x + y + (1 - t)\delta - y) \\
&\quad + (1 - v')(f(\phi_A(v)) - f(\phi_A(v')) + g(\phi_B(v)) - g(\phi_B(v'))) \\
&= -v'(\phi_A(v') - \phi_A(v) + \phi_B(v') - \phi_B(v)) \\
&\quad + (1 - v')(f(\phi_A(v)) - f(\phi_A(v')) + g(\phi_B(v)) - g(\phi_B(v'))) \\
&= \boldsymbol{v}' \cdot \phi_A(v) - \boldsymbol{v}' \cdot \phi_A(v') + \boldsymbol{v}' \cdot \phi_B(v) - \boldsymbol{v}' \cdot \phi_B(v') \\
&= \mathrm{divloss}(v, v'; A) + \mathrm{divloss}(v, v'; B)
\end{aligned}$$

A.14 Equation 22

$$\begin{aligned}
h'(x) &= -(tf'(a+tx) + (1-t)g'(b+(1-t)x)\\
&= -\left(t\frac{v}{(1-v)} + (1-t)\frac{v}{(1-v)}\right)\\
&= -\frac{v}{(1-v)}\\
\delta &= t\delta + (1-t)\delta\\
\delta &= \phi_A(v') - \phi_A(v) + \phi_B(v') - \phi_B(v)\\
\text{linslip}_X(v, v'; A||B) &= -(1-v')(-\delta h'(x) + h(x+\delta) - h(x))\\
&= -(1-v')\left(-\delta\frac{v}{1-v} + h(x+\delta) - h(x)\right)\\
&= \left(\frac{1-v'}{1-v}\right)(\delta v - (1-v)(h(x+\delta) - h(x)))\\
&= \left(\frac{1-v'}{1-v}\right)(\delta v - (1-v)(f(a) - f(a+t(x+\delta))\\
&\qquad + g(b) - g(b+(1-t)(x+\delta))\\
&\qquad - (f(a) - f(a+tx) + g(b) - g(b+(1-t)x))))\\
&= \left(\frac{1-v'}{1-v}\right)(\delta v + (1-v)(f(a+t(x+\delta)) - f(a+tx)\\
&\qquad + g(b+(1-t)(x+\delta)) - g(b+(1-t)x)))\\
&= \left(\frac{1-v'}{1-v}\right)(\delta v + (1-v)(f(\phi_A(v')) - f(\phi_A(v)) + g(\phi_B(v')) - g(\phi_B(v))))\\
&= \left(\frac{1-v'}{1-v}\right)(v(\phi_A(v') - \phi_A(v) + \phi_B(v') - \phi_B(v))\\
&\qquad + (1-v)(f(\phi_A(v')) - f(\phi_A(v)) + g(\phi_B(v')) - g(\phi_B(v))))\\
&= \left(\frac{1-v'}{1-v}\right)(v(\phi_A(v') - \phi_A(v)) + (1-v)(f(\phi_A(v')) - f(\phi_A(v)))\\
&\qquad + v(\phi_B(v') - \phi_B(v)) + (1-v)(g(\phi_B(v')) - g(\phi_B(v))))\\
&= \left(\frac{1-v'}{1-v}\right)(\boldsymbol{v}\cdot\Phi_A(v') - \boldsymbol{v}\cdot\Phi_A(v')) + (\boldsymbol{v}\cdot\Phi_B(v') - \boldsymbol{v}\cdot\Phi_B(v'))\\
&= \text{linslip}_X(v, v'; A) + \text{linslip}_X(v, v'; B).
\end{aligned}$$

A.15 Equation 23

$$\begin{aligned}
\text{linslip}_X(v, v'; \alpha A) &= \boldsymbol{v}'\cdot(\alpha x, \alpha f(x)) - \boldsymbol{v}'\cdot(\alpha x', \alpha f(x'))\\
&= \alpha(\boldsymbol{v}'\cdot\Phi(v) - \boldsymbol{v}'\cdot\Phi(v')\\
&= \alpha\,\text{linslip}_X(v, v'; A).
\end{aligned}$$

B Appendix: Proofs

B.1 Theorem 2

Proof. Let $h(v) = \boldsymbol{v}\cdot\Phi(v) = v\phi(v) + (1-v)f(\phi(v))$ be the capitilization at v. Note that h is a continuous function on the compact set $[0, 1]$ which guarantees the existence of the maximum. Let v^* be the point where the maximum occurs. The first derivative is

$$h'(v) = \phi(v) + v\phi'(v) + (1-v)f'(\phi(v))\phi'(v) - f(\phi(v))$$
$$\phi(v) + v\phi'(v) + (1-v)\frac{-v}{1-v}\phi'(v) - f(\phi(v))$$
$$= \phi(v) - f(\phi(v))$$

The second derivative is

$$h''(v) = \phi'(v) - f'(\phi(v))\phi'(v)$$
$$= \phi'(v)(1 - \frac{-v}{1-v})$$
$$= \phi'(v)(\frac{1-v+v}{1-v}) = \frac{\phi'(v)}{1-v}$$

Now take a derivative with respect to v of

$$f'(\phi(v)) = \frac{-v}{1-v}$$
$$\phi'(v) = \frac{-1}{(1-v)^2 f''(\phi(v))} < 0.$$

Recall that f is strictly convex, so $f''(\phi(v)) > 0$ for all $v \in (0,1)$.

We can then write the second derivative of the capitalization as

$$h''(v) = \frac{-1}{(1-v)^3 f''(\phi(v))} < 0$$

Thus h is strictly concave so the maximum is unique. Finally, the first-order conditions tell us that $h'(v^*) = 0$ or $\phi(v^*) = f(\phi(v^*))$. ◀

B.2 Lemma 3

Proof. Let $g = f^{-1} = f$.

$$f'(y) = g'(y) = \frac{1}{f'(f^{-1}(y))} = \frac{1}{f'(f(y))} = \frac{1}{f'(x)} = -\frac{(1-v)}{v} = \frac{-(1-v)}{1-(1-v)}$$

Thus $y = \phi(1-v)$. ◀

B.3 Theorem 4

Proof. From the proof of Theorem 2 we know $\phi'(v) < 0$ for $v \in (0,1)$, so $\phi(v)$ is strictly decreasing. Applying Theorem 2 and Lemma 3 tells us that capitalization is maximized when $\phi(v) = x = y = \phi(1-v)$. Because ϕ is strictly decreasing, the only way $\phi(v) = \phi(1-v)$ is if $v = 1-v$ or $v = \frac{1}{2}$. ◀

C Appendix: Minimizing Divergence Loss Exposure

Let $A := (x, f(x))$ be an AMM currently in state $(a, f(a))$, the stable state for $(v, 1-v)$. Define an X-*partition* to be a sequence of values $\{x_i\}_{i=1}^{\infty}$ in $\mathbb{R}_{>0}$ such that $x_1 = a$, $x_i < x_{i+1}$, and $\lim_{n\to\infty} x_n = \infty$.

Given a partition $P_X = \{x_i\}_{i=1}^{n}$, define the *total divergence loss* with respect to that partition as

$$\text{divloss}(P_X; A) = \sum_{i=1}^{\infty} \text{divloss}^*(x_i, x_{i+1}; A)$$

Writing this out explicitly gives

$$\begin{aligned}
&\sum_{i=1}^{\infty} \mathrm{divloss}^*(x_i, x_{i+1}; A) \\
&= \sum_{i=1}^{\infty} \boldsymbol{v}_{i+1} \cdot (\Phi(\boldsymbol{v}_i) - \Phi(\boldsymbol{v}_{i+1})) \\
&= \sum_{i=1}^{\infty} \boldsymbol{v}_{i+1} \cdot ((\phi(v_i), f(\phi(v_i))) - (\phi(v_{i+1}), f(\phi(v_{i+1})))) \\
&= \sum_{i=1}^{\infty} (v_{i+1}, 1 - v_{i+1}) \cdot (\phi(v_i) - \phi(v_{i+1}), f(\phi(v_i)) - f(\phi(v_{i+1}))) \\
&= \sum_{i=1}^{\infty} v_{i+1}(\phi(v_i) - \phi(v_{i+1})) - v_{i+1}(f(\phi(v_i)) - f(\phi(v_{i+1}))) + \sum_{i=1}^{\infty} f(\phi(v_i)) - f(\phi(v_{i+1})) \\
&= \sum_{i=1}^{\infty} v_{i+1}(\phi(v_i) - \phi(v_{i+1})) - v_{i+1}(f(\phi(v_i)) - f(\phi(v_{i+1}))) + f(\phi(v_1)) \\
&= \sum_{i=1}^{\infty} v_{i+1}[\phi(v_i) - \phi(v_{i+1}) + f(\phi(v_{i+1})) - f(\phi(v_i))] + f(\phi(v_1))
\end{aligned}$$

Note that each term $\phi(v_i) - \phi(v_{i+1}) + f(\phi(v_{i+1})) - f(\phi(v_i)) \leq 0$. We also know that $v_1 \geq v_{i+1}$ for each i. This gives us the upper bound

$$\sum_{i=1}^{\infty} v_{i+1}[\phi(v_i) - \phi(v_{i+1}) + f(\phi(v_{i+1})) - f(\phi(v_i))] + f(\phi(v_1)) \leq f(\phi(v_1)) = f(x)$$

and the lower bound

$$\begin{aligned}
&\sum_{i=1}^{\infty} v_{i+1}[\phi(v_i) - \phi(v_{i+1}) + f(\phi(v_{i+1})) - f(\phi(v_i))] + f(\phi(v_1)) \\
&\geq v_1 \sum_{i=1}^{\infty} [\phi(v_i) - \phi(v_{i+1}) + f(\phi(v_{i+1})) - f(\phi(v_i))] + f(\phi(v_1)) \\
&= v_1[\phi(v_1) - f(\phi(v_1))] + f(\phi(v_1)) \\
&= v_1\phi(v_1) + (1 - v_1)f(\phi(v_1)) \\
&= \mathrm{cap}(v; A)
\end{aligned}$$

Simply put

$$\mathrm{cap}(v; A) \leq \mathrm{divloss}(P_X; A) \leq f(a)$$

How tight can this lower bound get? Well, let v^* be the point where $\mathrm{cap}(v; A)$ is maximized. If we let $x^* = \phi(v^*)$, then we know that $f(x^*) = x^*$. We know that $\mathrm{cap}(v; A) = v^*x^* + (1 - v^*)f(x^*) = f(x^*)$. But if $x = x^*$ then this means

$$f(x^*) \leq \mathrm{divloss}(P; A) \leq f(x^*)$$

which is entirely independent of the chosen partition P_X, so

$$f(x^*) \leq \mathrm{divloss}(A) \leq f(x^*)$$

or $\mathrm{divloss}(A) = f(x^*)$. Additionally, total loss is conserved even if we modify the AMM A for $x > x^*$. No matter how you choose to drain asset type Y by depositing asset type X, in the end you will drain all of $f(x^*)$ if you deposit an infinite amount of X.

We get a similar result when trading along the Y-axis. Define a *Y-partition* to be a sequence of elements $\{x_i\}_{i=1}^{\infty}$ in $\mathbb{R}_{>0}$ such that $y_1 = f(a)$, $y_i \leq y_{i+1}$, and $\lim_{n\to\infty} y_n = \infty$.

Let $P_Y = \{y_i\}_{i=1}^{\infty}$ be a Y-partition, $g = f^{-1}(x)$. and $y_i = f(x_i)$ so $x_i = f^{-1}(y_i) = g(y_i)$. Thus $\Phi(\boldsymbol{v}_i) = (x_i, f(x_i)) = (g(y_i), f \circ g(y_i))$ where $f \circ g(y_i) = f \circ f^{-1}(y_i) = y_i$. That is, $\Phi(\boldsymbol{v}_i) = (g(y_i), y_i)$. The total cost with respect to this partition is

$$\text{divloss}(P_Y; A) = \sum_{i=1}^{\infty} \text{divloss}^*(y_i, y_{i+1}; A)$$

For symmetry, define $v_i' = 1 - v_i$.

$$\begin{aligned}
\sum_{i=1}^{\infty} \text{divloss}^*(y_i, y_{i+1}; A) &= \sum_{i=1}^{\infty} \boldsymbol{v}_{i+1} \cdot (\Phi(\boldsymbol{v}_i) - \Phi(\boldsymbol{v}_{i+1)}) \\
&= \sum_{i=1}^{\infty} \boldsymbol{v}_{i+1} \cdot (g(y_i) - g(y_{i+1}), y_i - y_{i+1}) \\
&= \sum_{i=1}^{\infty} (1 - v_{i+1}')(g(y_i) - g(y_{i+1}) + v_{i+1}'(y_i - y_{i+1}) \\
&= \sum_{i=1}^{\infty} v_{i+1}'[y_i - y_{i+1} + g(y_{i+1}) - g(y_i)] + \sum_{i=1}^{\infty} (g(y_i) - g(y_{i+1}) \\
&= \sum_{i=1}^{\infty} v_{i+1}'[y_i - y_{i+1} + g(y_{i+1}) - g(y_i)] + a
\end{aligned}$$

Note that $g(y_i) - g(y_{i+1})) + y_{i+1} - y_i \leq 0$ and $v_1' \geq v_{i+1}'$ for each i. We now get the upper bound

$$\sum_{i=1}^{\infty} v_{i+1}'[y_i - y_{i+1} + g(y_{i+1}) - g(y_i)] + a \leq a$$

and the lower bound

$$\begin{aligned}
\sum_{i=1}^{\infty} v_{i+1}'[y_i - y_{i+1} + g(y_{i+1}) - g(y_i)] + a &\geq v_1' \sum_{i=1}^{\infty} y_i - y_{i+1} + g(y_{i+1}) - g(y_i) + a \\
&= v_1'(y_1 - g(y_1)) + a \\
&= (1 - v_1)(f(a) - a) + a \\
&= (1 - v_1)f(a) + v_1 a = \text{cap}(v; A)
\end{aligned}$$

So again we get the bounds

$$\text{cap}(v; A) \leq \text{divloss}(P_Y; A) \leq a$$

Similar to the X-axis case this inequality is tight if $y_1 = f(x^*)$ and $\text{divloss}(P_Y; A) = a$. Thus we do get a loss conservation result if we start at $(x^*, f(x^*))$. Namely

$$\text{divloss}(P_X; A) + \text{divloss}(P_Y; A) = x^* + f(x^*) = 2x^* = 2\,\text{cap}(v^*; A)$$

The valuation $\boldsymbol{v}^*$ thus corresponds to the AMM state where half of the wealth may be lost to X trades and half can be lost to Y trades.

D Mathematica Code

This section shows the Mathematica scripts used to generate Figure 3.

```
g[x_] := InverseFunction[f][x];
\[Phi][v_] := InverseFunction[f'][-v/(1-v)];
 divloss [v1_,v2_] :=
     v2 \[Phi][v1] + (1-v2) f[\[Phi][v1]] - (v2 \[Phi][v2] + (1-v2) f[\[Phi][v2]]);

 linslipx [v1_,v2_] :=
     ((1-v2)/(1-v1)) (v1 \[Phi][v2] - v1 \[Phi][v1]
         + (1-v1) f[\[Phi][v2]] - (1-v1) f[\[Phi][v1]]);

 linslipy [v1_,v2_] :=
     (v2/v1) ((1-v1) f[\[Phi][v2]] - (1-v1) f[\[Phi][v1]]
         + v1 \[Phi][v2] - v1 \[Phi][v1]);

loadx[v1_,v2_] := divload[v1,v2]  linslipx [v1,v2];
loady[v1_,v2_] := divloss[v1,v2]  linslipy [v1,v2];

exploadx[v_,\[Alpha]1_,\[Alpha]2_] :=
     PDF[BetaDistribution[\[Alpha]1, \[Alpha]2]][v] loadx[1/2,v];
exploady[v_,\[Alpha]1_,\[Alpha]2_] :=
     PDF[BetaDistribution[\[Alpha]1, \[Alpha]2]][v] loady[1/2,v];
expload[\[Alpha]1_, \[Alpha]2_] :=
 NIntegrate[exploadx[v, \[Alpha]1, \[Alpha]2], {v, 0, 1/2}] +
  NIntegrate[exploady[v, \[Alpha]1, \[Alpha]2], {v, 1/2, 1}]

 expslipx [v_,\[Alpha]1_,\[Alpha]2_] :=
     PDF[BetaDistribution[\[Alpha]1, \[Alpha]2]][v]  linslipx [1/2,v];
 expslipy [v_,\[Alpha]1_,\[Alpha]2_] :=
     PDF[BetaDistribution[\[Alpha]1, \[Alpha]2]][v]  linslipy [1/2,v];
 expslip [\[Alpha]1_, \[Alpha]2_] :=
 NIntegrate[ expslipx [v, \[Alpha]1, \[Alpha]2], {v, 0, 1/2}] +
  NIntegrate[ expslipy [v, \[Alpha]1, \[Alpha]2], {v, 1/2, 1}]

exploadx[v_,\[Alpha]1_,\[Alpha]2_] :=
     PDF[BetaDistribution[\[Alpha]1, \[Alpha]2]][v] divloss [1/2,v];
exploady[v_,\[Alpha]1_,\[Alpha]2_] :=
     PDF[BetaDistribution[\[Alpha]1, \[Alpha]2]][v] divloss [1/2,v];
expload[\[Alpha]1_, \[Alpha]2_] :=
 NIntegrate[exploadx[v, \[Alpha]1, \[Alpha]2], {v, 0, 1/2}] +
  NIntegrate[exploady[v, \[Alpha]1, \[Alpha]2], {v, 1/2, 1}]

Plot3D[expload[\[Alpha]1, \[Alpha]2], {\[Alpha]1, 1, 4}, {\[Alpha]2,
  1, 4}, PlotRange -> All, MeshFunctions -> {#3 &}]
```