BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

INCENTIVE STRATEGIES FOR TRANSACTION-FEE REGIME OF PROOF-OF-WORK BLOCKCHAINS

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR RASTISLAV BUDINSKÝ
AUTOR PRÁCE

SUPERVISOR Ing. IVAN HOMOLIAK, Ph.D.
VEDOUCÍ PRÁCE

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Bachelor's Thesis Assignment

148520

Institut:	Department of Intelligent Systems (UITS)
Student:	**Budinský Rastislav**
Programme:	Information Technology
Specialization:	Information Technology
Title:	**Incentive Strategies for Transaction-Fee Regime of Proof-of-Work Blockchains**
Category:	Modelling and Simulation
Academic year:	2022/23

Assignment:

1. Get familiar with existing Proof-of-Work consensus protocols and their incentive schemes. Focus on the analysis of the drawbacks of such schemes, especially in the case of the transaction-fee regime.
2. Study existing simulation testbeds for Proof-of-Work consensus protocols and compare their properties.
3. Propose a new incentive scheme aimed at the transaction-fee regime.
4. Implement a proof-of-concept model of the proposed scheme and perform its simulation under various conditions.
5. Discuss the results and state the recommendations for various use cases.

Literature:

- Carlsten, Miles, et al. "On the instability of bitcoin without the block reward." *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016.
- Homoliak, Ivan, et al. "The security reference architecture for blockchains: Towards a standardized model for studying vulnerabilities, threats, and defenses." *arXiv preprint arXiv:1910.09775* (2019).

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Homoliak Ivan, Ing., Ph.D.**
Head of Department:	Hanáček Petr, doc. Dr. Ing.
Beginning of work:	1.11.2022
Submission deadline:	10.5.2023
Approval date:	3.11.2022

Abstract

In this thesis, we review the undercutting attacks in the transaction-fee-based regime of Proof-of-Work (PoW) blockchains with the longest chain fork-choice rule. Next, we focus on the problem of fluctuations in mining revenue and the mining gap – i.e., a situation, in which the immediate reward from transaction fees does not cover miners' expenditures.

To mitigate these issues, we propose a solution that splits transaction fees from a mined block into two parts – (1) an instant reward for the miner of a block and (2) a deposit sent to one or more Fee-Redistribution Contracts ($\mathcal{FRC}$s) that are part of the consensus protocol. At the same time, these redistribution contracts reward the miner of a block with a certain fraction of the accumulated funds of the incoming fees over a predefined time. This setting enables us to achieve several interesting properties that are beneficial for the incentive stability and security of the protocol.

With our solution, the fraction of Default-Compliant miners who strictly do not execute undercutting attack is lowered from the state-of-the-art result of 66% to 30%.

Abstrakt

V tejto práci sa zaoberáme tkz. undercutting útokmi v blockchainoch založených na transakčných poplatkoch (Proof-of-Work, PoW) s pravidlom voľby najdlhšieho reťazca pri vzniknutí forkov. Ďalej sa zameriame na problém kolísania výnosov z ťažby a tkz. mining gap – teda situáciu, v ktorej okamžitá odmena z transakčných poplatkov nepokryje výdaje ťažiarov.

Na zmiernenie týchto problémov navrhujeme riešenie, ktoré rozdeľuje transakčné poplatky z vyťaženého bloku na dve časti – (1) okamžitú odmenu pre ťažiara bloku a (2) vklad odoslaný do jedného alebo viacerých kontraktov na prerozdelenie poplatkov nazvaných poplatky-prerozdeľovacie kontrakty (Fee-Redistribution Contracts – $\mathcal{FRC}$s), ktoré sú súčasťou konsenzuálneho protokolu. Tieto poplatky-prerozdeľovacie kontrakty zároveň odmeňujú minera bloku určitým zlomkom naakumulovaných prostriedkov z prichádzajúcich poplatkov za vopred stanovený čas. Toto nastavenie nám umožňuje dosiahnuť niekoľko zaujímavých vlastností, ktoré sú prospešné pre stabilnú incentívu pre minerov a bezpečnosť protokolu.

S naším riešením sa zlomok Default-Compliant minerov, ktorí priamo nevykonávajú undercutting útok, zníži z pôvodného výsledku 66% na 30%.

Keywords

Bitcoin, blockchain, block, mining, PoW, Proof-of-Work, transaction-fee regime, undercutting attack, mining gap, mining revenue fluctuation, fee-redistribution contracts

Klíčová slova

Bitcoin, blockchain, blok, ťaženie, PoW, Proof-of-Work, režimy s poplatkami, undercutting útok, mining gap

Reference

BUDINSKÝ, Rastislav. *Incentive Strategies for Transaction-Fee Regime of Proof-of-Work Blockchains.* Brno, 2023. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Ivan Homoliak, Ph.D.

Incentive Strategies for Transaction-Fee Regime of Proof-of-Work Blockchains

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Mr. Ing. Ivan Homoliak Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

....
Rastislav Budinský
May 5, 2023

Acknowledgements

I would like to thank to my supervisor Ing. Ivan Homoliak Ph.D. for professional guidance, consultation and many valuable advices. I would also like to show my gratitude to my supervisor for great collaboration on a research paper, which preceded this thesis. Lastly I would to thank to Ergo core developer Alexander Cherpunoy for his initial inputs about this idea and Ing. Ivana Stančíková for consultancy and help with afford-mentioned preceding research paper we created together.

Contents

1 Introduction **4**

2 Extended Abstract **6**
2.1 Problémy, ktorým sa primárne venujeme 6
2.2 Navrhované riešenie 8
2.3 Výsledky riešenia 8
2.4 Príspevky 8

3 Preliminaries **10**
3.1 Blockchain technology 10
3.2 Mempool 12
3.3 Transaction 12
3.4 Block 13
3.5 Block reward 13
3.6 Total supply 14
3.7 Transaction fees 14
3.8 Proof of Work 15
3.9 Mining 16

4 Current Problems with Transaction-Fee-Based Regimes **17**
4.1 Fluctuation of miner's reward 17
4.2 Undercutting attack 18
4.3 Mining gap 19
4.4 Extisting solutions 20

5 Proposed Solution **22**
5.1 Overview 22
5.2 Prioritization of transactions with higher feerate 23
5.3 Fee-Redistribution Contracts 23
5.4 Example 24
5.5 Traditional way in transaction-fee-based regime 25
5.6 Initial setup of $\mathcal{FRC}$s contracts 25

6 Experiments **27**
6.1 Simulator 27
6.2 Experiment I – Different lengths of $\mathcal{FRC}$s 31
6.3 Experiment II – Multi-contract $\mathcal{FRC}$ 31
6.4 Experiment III – Effective length of $\mathcal{FRC}$ 33

6.5 Experiment IV – Historical data of miner's reward fluctuation 35
6.6 Experiment V – Undercutting attack with $\mathcal{FRC}$s used 37

7 Security Analysis and Discussion 40
7.1 Contract-drying attack 40
7.2 Possible improvements 41
7.3 Epoch like $\mathcal{FRC}$s 41
7.4 Adjustment of mining difficulty 41
7.5 Allowing soft fork implementation 42
7.6 So called out-of-band fees 42
7.7 Utilization of Fee-Redistribution Contracts 43

8 Conclusion 44

Bibliography 46

A Fee-Redistribution Contracts on Ergo Blockchain 48
A.1 Brief introduction into Ergo 48
A.2 Smart contract overview 48
A.3 Implementation 49

B Contents of Included Storage Device 51

List of Figures

3.1 Blockchain cryptographically linked. Image source [19]. 12

3.2 Illustration of pending transactions in mempool and transactions in a block. Image source [1]. 13

3.3 Example of Bitcoin transaction with fee for miner as “Unclaimed”. Image source [2]. 14

3.4 Example of Bitcoin block. Image source [3]. 15

3.5 Process of mining in PoW blockchain. Image source [15]. 16

4.1 Fluctuation in transaction fees Q4 2017. Data provided by [23]. 18

4.2 Undercutting attack from the original paper [23]. 19

5.1 Overview of our solution. 23

6.1 Experiment I investigating various $\mathbb{C}s$ and λs of a single $\mathcal{FRC}$, where $\mathcal{FRC}^1.\lambda = 2016$ and $\mathcal{FRC}^2.\lambda = 5600$. *Fees in mempool* show the total value of fees in the mempool able to be mined (i.e., representing the baseline). *Block Value* is the actual reward a miner received in block B as a sum of the fees he obtained directly (i.e. $\mathbb{M} * B.fees$) and the reward he got from $\mathcal{FRC}$ (i.e., $nextClaim_{[H]}$). *Expected income from Contract* represents the reward of a miner obtained from $\mathcal{FRC}$ (i.e., $nextClaim_{[H]}$). 32

6.2 Experiment II investigating various $\mathbb{C}$s in the setting with multiple $\mathcal{FRC}$s with their corresponding $\lambda = \{1008, 2016, 4032, 8064\}$ and $\rho = \{0.07, 0.14, 0.28, 0.51\}$. $\partial Claim$s represents contributions of individual $\mathcal{FRC}$s to the total reward of the miner (i.e., its *nextClaim* component). 34

6.3 Experiment II – multiple $\mathcal{FRC}$s using various distributions of ρ and their impact on $\partial Claim$, where $\mathbb{C} = 0.7$. 35

6.4 Experiment III comparing 4 $\mathcal{FRC}$s and 1 $\mathcal{FRC}$, both configurations having the same effective_λ. 36

6.5 Experiment IV – Direct recreation of Figure 4.1 into a scenario against our solution with $\mathcal{FRC}$s. 37

6.6 Stacked area chart showing equilibrium distributions of different mining strategies. The number of DEFAULT-COMPLIANT miners in our $\mathcal{FRC}$ approach is ∼30% (in contrast to ∼ 66% of [5]). 39

Chapter 1

Introduction

Cryptocurrencies provide block rewards to incentivize miners in producing new blocks. Transaction fees also contribute to the revenue of miners, motivating them to include transactions in blocks. Miners maximize their profits by prioritizing the transactions with the highest ratio of fees to transaction size. In Bitcoin and its numerous clones [13], the block reward is divided by a factor of two approx. every four years (i.e., after every 210k blocks), which will eventually result in a pure transaction-fee-based regime with no income for miners from block reward.

Not much thought was given to this problem in the Bitcoin whitepaper [18], citing the author, "*Once a predetermined number of coins have entered circulation, the incentive can transition entirely to transaction fees and be completely inflation free.*"

Before 2016, there was also a belief that the dominant source of the miners' income does not impact the security of the blockchain. However, Carlsten et al. [5] pointed out the effects of the high variance of the miners' revenue per block caused by exponentially distributed block arrival time in transaction-fee-based protocols. The authors showed that *undercutting* (i.e., forking) a wealthy block is a profitable strategy for a malicious miner. Nevertheless, Daian et al. [7] showed that this attack is viable even in blockchains containing traditional block rewards due to front-running competition of arbitrage bots who are willing to extremely increase transaction fees to earn Maximum Extractable Value profits.

In this paper, we focus on mitigation of the undercutting attack in transaction-fee-based regime of PoW blockchains. We also discuss related problems present (not only) in transaction-fee-based regime. In particular, we focus on minimizing the mining gap [5, 21], (i.e., the situation, where the immediate reward from transaction fees does not cover miners' expenditures) as well as balancing significant fluctuations in miners' revenue.

To mitigate these issues, we propose a solution that splits transaction fees from a mined block into two parts – (1) an instant reward for the miner and (2) a deposit sent into one or more fee-redistribution contracts ($\mathcal{FRC}$s). At the same time, these $\mathcal{FRC}$s reward the miner of a block with a certain fraction of the accumulated funds over a fixed period of time (i.e., the fixed number of blocks). This setting enables us to achieve several interesting properties that are beneficial for the stability and security of the protocol.

Contributions

In detail, our contributions are as follows:

1. We propose an approach that normalizes the mining rewards coming from transaction fees by one or more $\mathcal{FRC}$s that perform moving average on a certain portion of the transaction fees.

2. We evaluate our approach using various fractions of the transaction fees from a block distributed between a miner and $\mathcal{FRC}$s. We experiment with the various numbers and lengths of $\mathcal{FRC}$s, and we demonstrate that usage of multiple $\mathcal{FRC}$s of various lengths has the best advantages mitigating the problems we are addressing; however, even using a single $\mathcal{FRC}$ is beneficial.

3. We demonstrated that with our approach, the mining gap can be minimized since the miners at the beginning of the mining round can get the reward from $\mathcal{FRC}$s, which stabilizes their income.

4. We conduct a simple experiment on historical real-world data as example to demonstrate the benefit of using $\mathcal{FRC}$s compared to baseline without our solution to directly show the impact of our approach.

5. We empirically demonstrate that using our approach, the threshold of Default-Compliant miners who strictly do not execute undercutting attack is lowered from 66% (as reported in the original work [5]) to 30%.

Organization

This thesis is organized as follows. Chapter 3 Defines basic terms used in this thesis In Chapter 4 defines the problem that we are dealing with. In Chapter 5, we describe our approach. In Chapter 6, we conduct the experiments to evaluate our approach in terms of lowering the profitability of undercutting attacks. In Chapter 7 we conduct security analysis of our solution and discuss possible improvements and other factors. Finally, in Chapter 8 we summarize our findings of our proposed solution in this thesis.

Chapter 2

Extended Abstract

V tejto bakalárskej práci nadväzujeme na problémy skúmané vo vedeckom článku z roku 2016, ktorý napísali Carlsten et al. [5]. Problémy, ktorými sa zaoberajú v tejto práci a rovnako aj my, sú demonštrované na Proof-of-Work blockchaine Bitcoin. Jedná sa o problémy podrezávacích útokov (ang. undercutting attack), medzera v príjmoch z odmien (ang. minig gap), a fluktuácie odmeny pre minerov. Tieto problémy vyplývajú zo skutočnosti, že každá kryptomena, s maximálnym obmedzeným počtom mincí v obehu (ang. capped supply), ako je Bitcoin, bude eventuálne závislá len na odmenách z transakčných poplatkov. Toto bude jediný spôsob motivácie pre ťažiarov (ang. miner) zabezpečovať sieť.

V súčasnej situácii sú mineri motivovaní stále sa zmenšujúcou odmenou z bloku, tzv. block reward. V budúcnosti tomu však tak nebude. V relatívne blízkej budúcnosti sa očakáva, že block reward z blokov bude menší ako z transakčných poplatkov. Približne od roku 2140 už nebudú žiadne block rewardy v Bitcoine. Treba podotknúť, že problémy prezentované v tejto výskumnej práci, sa síce prejavujú už teraz, avšak ich skutočný dopad môže byť nižší, ako sa pôvodne predpokladalo.

To však neznamená, že tieto problémy sú menej dôležité. Skôr to znamená, že v reálnych systémoch sa nemusia vyskytovať tak často alebo mať taký veľký vplyv, ako sa pôvodne predpokladalo, keď bol článok prvýkrát zverejnený. Existujú aj iné výskumné práce, ktoré zvyčajne navrhujú riešenie len jedného z prezentovaných problémov. Naviac niektoré z týchto riešení skôr navrhujú preventívny prístup, ktorý ale nepovažujeme za dostatočne prísny. Navrhujeme riešenie, ktoré by malo riešiť viacero z prezentovaných problémov. Zároveň riešenie je jednoduché na pochopenie a ľahko začleniteľné do súčasných a budúcich blockchainov.

2.1 Problémy, ktorým sa primárne venujeme

Carlsten et al. [5] skúmali a opísali viacero problémov. Poskytli pádne argumenty za týmito problémami, ktoré demonštrovali na simulátore, vytvorenom v tejto práci. Spolu s kombináciou s novými dátami ukazujeme viac príkladov z reálneho sveta a testujeme s nimi naše riešenie. V tejto práci sa zameriame na zmiernenie účinkov nasledujúcich troch problémov zo spomínanej výskumnej práce.

2.1.1 Fluctuation of miner's reward

Kolísanie odmien minera je situácia, kedy sa v priebehu určitého časového úseku odmeny z transakcií výraznejšie menia, čo vedie ku kolísaniu odmien medzi jednotlivými vyťaženými blokmi. To priamo ovplyvňuje základné predpoklady teórie hier, čo vyúsťuje k narušeniu celkovej bezpečnosti systému. Chápeme, že tento problém možno pozorovať na báze niekoľkých blokov, ide však o chaotické správanie, ktoré je obtiažne analyzovať. Preto túto variantu v našej práci neuvažujeme.

Z tohto dôvodu sa radšej zameriame na predvídateľnejšie a opakovateľnejšie časové rámce ako napríklad vrámci dňa, týždňa či mesiaca, čo možno považovať za opodstatnené obavy, ako ukazujú tieto grafy z roku 2017 [23]. Z grafov môžeme vyčítať rozdiely až viac ako 15% v ziskovosti minerov medzi jednotlivými, po sebe nasledujúcimi dňami v týždni. Ďalším príkladom môže byť očakávané zvýšenie dopytu počas sviatkov či prázdnin.

Radi by sme zmiernili vplyv týchto výkyvov, aby sme zachovali predvídateľnejšiu ziskovosť minerov a vytvorili prostredie, z ktorého profituje aj konečný užívateľ.

2.1.2 Undercutting attack

Tento typ útoku bol prvýkrát predstavený v [5] a bol podporený vykonanou simuláciou v tom istom článku. Pre jednoduchosť is uvedemie 3 typy minerov.

Poctivý miner, ktorý dodržiava protokolárne pravidlá. Ťaží na najvyššom bloku a neuprednostňuje maximalizáciu vlastnej ziskovosti.

Celkom poctivý miner, ktorý dodržiava väčšinu pravidiel protokolu, ale snaží sa maximalizovať svoju ziskovosť. Týmto porušuje napr. pravidlo najdlhšieho reťazca (ang. longest-chain rule).

Posledným typom je **útočný miner**, ktorý nedodržiava viaceré pravidlá protokolu. Zároveň aktívne útočí na náš systém tým, že sa snaží narušiť celkový konsenzus siete. Avšak stále vytvára platné bloky. Tento miner sa snaží pomocou útokov maximalizovať svoju vlastnú ziskovosť z ťažby.

Útočný miner sa pokúša znovu vyťažiť už existujúci blok takým spôsobom, že novovytvorený blok obsahuje len podmnožinu zo všetkých transakcií, zahrnutých v pôvodnom bloku. Ďalej ponechá niektoré transakcie nevyťažené a tým motivuje ďalšieho minera ťažiť na práve jeho bloku. To znamená, že ak má jeden blok veľa transakcií s vysokými poplatkami (ang. high feerate), t. j. vyšším pomerom poplatkov za bajt, útočný miner tento blok znovu vyťaží. Popri tom zahrnie iba podmnožinu všetkých týchto vysokohodnotných transakcií a zvyšok ponechá pre (celkom) poctivého minera, ktorý ich opätovne vyťaží. Týmto profitujú obidvaja mineri.

To priamo vedie k tomu, že často dochádza k rozdeľovaniu blockchainu (ang. fork) a odrádza poctivých minerov. Tým ich motivuje, aby prešli na celkom poctivých minerov. Častejšie forky vedú k dlhšiemu času medzi vyťaženými blokami (ang. mean block time). Toto negatívne ovplyvňuje skúsenosti koncového používateľa pri interakciách s blockchainom.

2.1.3 Mining gap

Tento problém robí nerentabilné pre minerov ťažiť hneď po vyťažení nového bloku, pretože tento blok zahŕňa väčšinu transakcií, z jeho tzv. mempoolu. Ak by nejaký miner práve teraz vyťažil nový blok, tento blok by obsahoval väčšinu naších transakcií. V našom mempoole by ostalo veľmi málo alebo žiadne transakcie. Vyťažením bloku v tomto momente by nám neprinieslo dostatočnú odmenu z transakcií, a teda príjmy by nepokryli naše výdavky. To znamená, že mineri sa môžu rozhodnúť neťažiť, kým nezhromaždia dostatok transakcií v ich mempoole, aby poplatky z týchto transakcií pokryli ich výdavky. Toto môže viesť k preskakovaniu medzi rôznymi protokolmi, aby si udržal zisk. Takéto skákanie je väčšinou problémom ťažiteľných protokolov odolných voči ASIC-om. Toto opäť vedie k zhoršenej skúsenosti pre koncového užívateľa.

2.2 Navrhované riešenie

Navrhujeme riešenie, ktoré rozdeľuje transakčné poplatky, ktoré by za normálnych okolností išli priamo minerovi bloku, na dve časti.

Prvá časť (1) ako okamžitá odmena pre minera bloku. Aktuálne iba (1) by zodpovedala súčasnej situácii ako napr. v Bitcoine. Zaisťuje, že miner je motivovaný uprednostňovať transakcie s vyššími poplatkami a zahrnúť ich do vyťaženého bloku.

Druhá časť (2) záloha zaslaná do jedného alebo viacerých tzv. poplatky-prerozdeľujúcich kontraktov (ang. Fee-Redistribution Contracts) ($\mathcal{FRC}$s), ktoré sú súčasťou konsenzuálneho protokolu. Cieľom týchto kontraktov je spriemerovať prijaté finančné prostriedky získané z (2) od minerov.

Takýto kontrakt zároveň odmieňa minera bloku určitým zlomkom naakumulovaných prostriedkov v kontrakte, teda sú spriemerované. Môžeme mať kombináciu viacerých kontraktov, kde by každý kontrakt dostal iný podiel z vyzbieranej časti (2) a prostriedky by priemeroval za iné obdobie. Pomer medzi (1) a (2) je jedno z nastavení, ktoré je potrebné ustanoviť. To vedie k viacerým výhodám, napríklad rôzne kontrakty sa môžu pokúsiť spriemerovať odmeny voči rôznym časovým rámcom a zmierniť či úplne odstrániť prezentované problémy za určitých rozumných predpokladov.

2.3 Výsledky riešenia

Môžeme si empiricky overiť, že naše riešenie výrazne zmiernilo problém tzv. mining gapu (popr. ho úplne odstránilo) správnym nastavením $\mathcal{FRC}$s tým, že vždy poskytneme kvázi garantovanú odmenu z kontraktov. To znamená, že miner by nemusel zastavovať ťažbu ani preskakovať medzi protokolmi, kým nebude jeho mempool primerane saturovaný transakciami.

Ďalej ponúka vlastnosť spriemerovania posuvným oknom (ang. sliding window). Týmto priamo znižujeme efekt problému kolísania príjmov minerov z ťažby. Dosahuje sa to spriemerovaním príjmov, teda zníženiu odchyliek, za vopred definované časové obdobia, ktoré je možné nastaviť v jednotlivých kontraktoch.

Nakoniec, s konzervatívne zvolenými parametrami $\mathcal{FRC}$s, sme znížili minimálny potrebný počet poctivých minerov. Teda bod, v ktorom už nie je undercutting útok pre útočného minera rentabilný, z pôvodných 66% na 30%. Čo vedie k zníženiu hranice tohto útoku o približne 55%.

2.4 Príspevky

Navrhli sme potenciálne riešenie pre Bitcoin a iné kryptomeny, ktoré majú obmedzený maximálny počet mincí v obehu. Tieto blockchainy budú eventuálne závislými iba na transakčných poplatkoch. Toto riešenie však nepodlieha len takýmto blockchainom, avšak na tieto blockchainy vplýva najviac. Poskytli sme riešenie, ktorého cieľom je zachovať obmedzený maximálny počet mincí v obehu, ale aby boli transakčné poplatky z ťažby predvídateľnejšie a udržateľnejšie. Toto riešenie je možno implementovať prostredníctvom soft-forku na mnohých súčasných blockchainoch. Toto vedie k oveľa príjemnejšiemu riešeniu na rozdiel od riešení založených na hard-forkoch. Definitívne nastavenie kontraktov však neposkytujeme a malo by to byť predmetom skúmania jednotlivých protokolov.

Naše riešenie nie je predmetom len pre prerozdeľovanie transakčných poplatkov ako demonštrujeme v Appendix A

Chapter 3

Preliminaries

Blockchain technology was firstly introduced in the original whitepaper from year 2008 [18] as a result of the 2008 financial crisis. The next year this technology was implemented with first Bitcoins, the native cryptocurrency coin of the Bitcoin blockchain, mined on January 3rd. 2009. Bitcoin was created with the intention to offer fully transparent option of transacting value besides the existing legacy financial system. Many new use cases were discovered taking the advantage of such system. Ever since then, thousands of alternative cryptocurrencies have emerged trying to solve many shortcomings of the design of Bitcoin and extending it with new features.

Besides alternative way of exchanging financial value in native coins, new generation blockchains offer exchange of custom assets, persistent storage or execution of code. The main benefit of adding these features to such system is it allows using them without the need of a middleman. This is achieved by utilizing modern cryptography primitives, game theory [22] in incentivizing miners or validators to ensure validity of transaction in these systems and many more fields.

3.1 Blockchain technology

Blockchain can be seen as an immutable shared database or as a public ledger. Blockchain at it's core is a structure holding a sequence of cryptographically linked blocks. Therefore, the name blockchain – chain of blocks. Each block is constructed based on the rules of the underlying protocol, where in a block we can usually find multiple records of transactions created by the participants of the system during a certain time period, timestamp etc.

A single transaction must contain some mandatory data, such as arbitrary user can verify the validity of such transaction. A block is then appended on top of the blockchain, and it cryptographically refers to the previous top of the blockchain. These references ensure a random block from the structure can not be altered or removed, as this would result in breaking the referencing and making a sequence of blocks, from this block, invalid.

A protocol, using such structure, then introduces an interface for defining the main structure of individual blocks, possible interactions by different groups of users with the protocol etc. resulting in system of users interacting using this protocol. In such systems, there are usually no overarching authorities[1] that would control the system. Rather, these systems rely on consensus rules, which are interpreted via consensus nodes [11] and are actively participating in the consensus.

[1] We are describing permissionless blockchains such as Bitcoin

Consensus nodes hold the whole copy of blockchain and can verify existing blocks and newly added blocks satisfy predefined conditions. There is no such a thing as a central copy of the blockchain, but rather the copy is obtained by interacting with consensus nodes. Furthermore, by obtaining the history of blockchain from consensus nodes we can expect differences between individual copies of blockchain of top block(s) as this is intended to be eventually consistent database so to speak.

From now we will be using the word blockchain interchangeably between the structure definition and as the system of users using a protocol, which uses blockchain structure at it's core.

The blockchain has usually the following properties:

Decentralized. The system does not have any central authority. This results in a system based on peer to peer architecture between consensus nodes. The blockchain copy is distributed across consensus nodes of the system, making it also **distributed**.

Permisionless. Any participant can join the network and can become any part of the system e.g. regular user creating transactions, consensus node etc. Permission is granted to any user, who wants to join the system [11].

Trustless. At the core, the system is built around cryptographic primitives. Meaning, a participant in such a system does not need to trust any particular user, but he can verify everything by himself.

Consensus. It is achieved by the consensus nodes in the system, interactions with other consensus nodes and following the protocol rules. This ensures a general agreement of the status of the blockchain structure.

Persistent. Once a transaction is in a block, that is part of the blockchain and fork of the chain does not happen in the near future, it will be persisted quasi forever.

Immutable. Nothing in the blockchain can be changed once it is generally accepted across consensus nodes in the blockchain structure. This is the result of blocks being cryptographically linked and referenced in a linear fashion in the blockchain structure. However, this immutability can be only considered after some time has passed, when we expect no forks to take place. This time is usually referred to as finality time.

Security. As long as the cryptographical assumptions of underlying cryptographical primitives hold true, the system can be considered secured. However, we would additionally need to ensure we have enough honest participants in this system.

Transparent. Any participant can access and verify transactions created in the blockchain by himself. All information is publicly visible to all users.

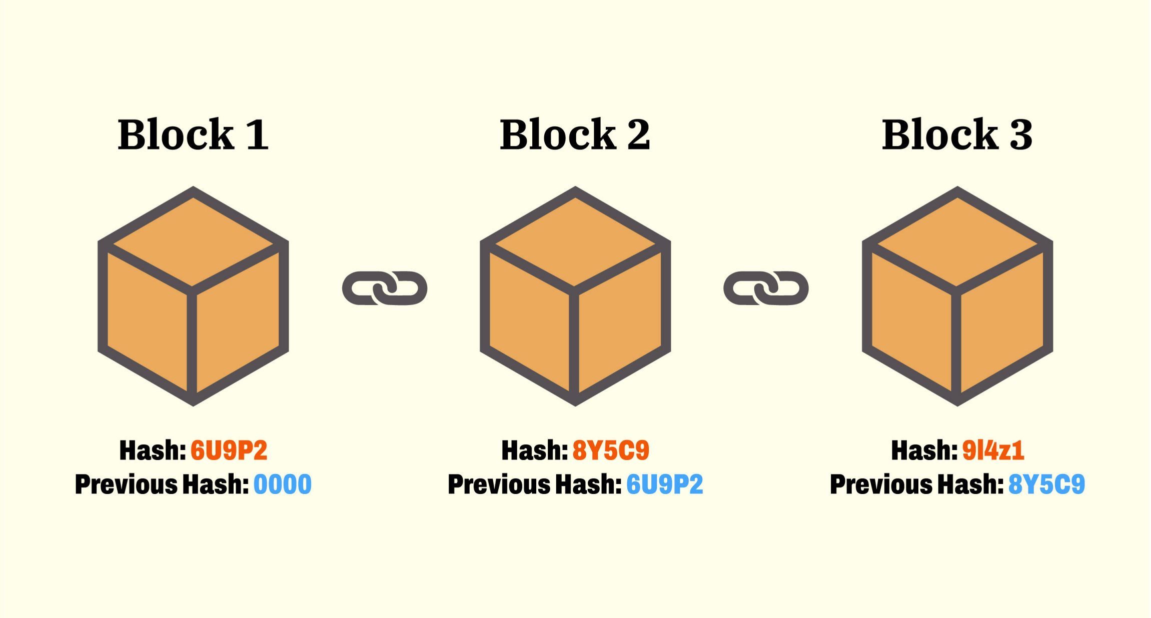


Figure 3.1: Blockchain cryptographically linked. Image source [19].

3.2 Mempool

A mempool is a pool of all transactions pending to be included in a block. Mempool is also distributed across consensus nodes (even though every consensus node has its own mempool), which receive new transactions, share the mempool with other peer consensus nodes and include a subset of these transactions when creating a new block.

Once a block is created all the transactions from this block are removed from the mempool to ensure the mempool contains only pending and unconfirmed transactions. Transactions included in a block have moved from unconfirmed status to confirmed status.

3.3 Transaction

A transaction can hold arbitrary data, but usually they are constructed from one of the functionalities of the underlying blockchain e.g. a transfer of coins or tokens, execution of script etc. In most blockchains, a mandatory transaction fee has to be paid to incentivize consensus nodes to accept our transaction and prevent spamming attacks.

Transactions are usually signed by their respective sender before they are sent to a consensus node, and before they are propagated from such node across the network. Signing transaction is done by asymmetric cryptography using private and public key pairs. We use private key to encrypt the data of the transaction and public key to verify the original content of the data. Or we can encrypt data by a public key and decrypt them only by the corresponding private key.

However, transactions can be executing some arbitrary scripts rather than just being signed by the sender as simple transfer transaction. Such scripts are called smart contracts [20] in blockchain terminology.

Once a transaction is added to a block and this block becomes part of the blockchain, it is usually considered confirmed. Moreover, in real blockchains occasional forks or other

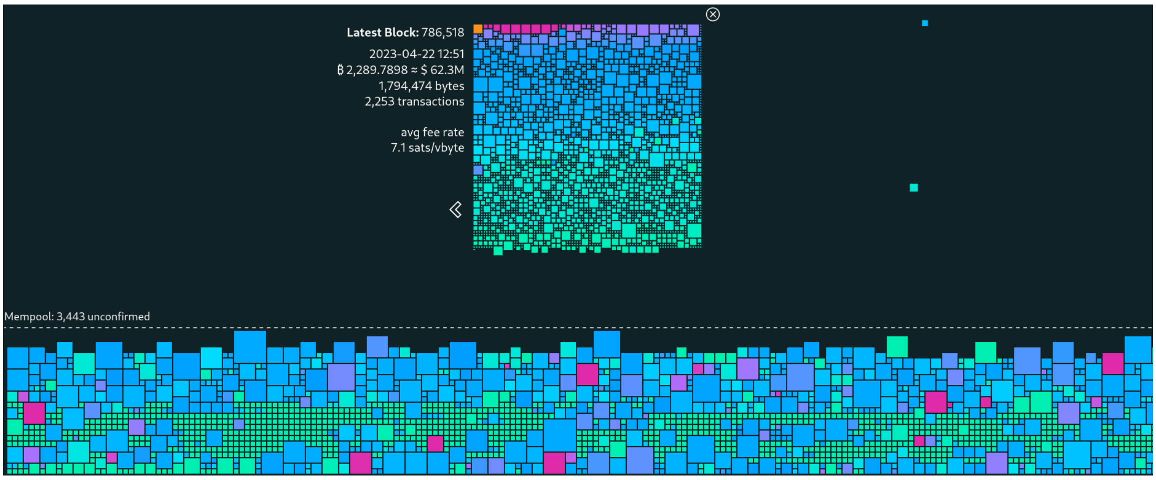


Figure 3.2: Illustration of pending transactions in mempool and transactions in a block. Image source [1].

actions might occur, which revert some of the top blocks resulting in the blocks not being part of the final blockchain anymore.

To consider transaction permanent we have to wait for the finality of a transaction [11], which is a recommended number of blocks, that should be built on top of the block with our transaction.

3.4 Block

A block consists of two parts – a header and a body. The header contains metadata such hash of the previous block, on top of which this one was created, Merkle tree root hash of all transactions, timestamp, nonce etc. The body contains the set of transactions included in this block and other related data. Put in other words a block aggregates all pending transaction during certain time frame from the mempool and makes these transactions confirmed.

However, a block usually has a limitation on how many transactions it can include arising from the block size (blockspace) i.e. the maximum size of bytes a block can have. This limitation makes the body of block a scarce resource.

For reference the size of Bitcoin block is 1 MB, which restrict the maximum number of transactions in such block to 12195 [6] transactions. Please note, the transaction count is done by assuming only simple transactions utilizing SegWit soft-fork.

3.5 Block reward

A block reward is generated with a new block being created. It usually creates new native coins of the protocol and rewards it to the creator of the block. This helps to incentivize miners or validators to secure the protocol, and it also helps with distributing the coins in more fair fashion between the participants of the blockchain.

Most blockchains rely on this scheme to ensure the security of the protocol by inviting miners or validators to validate the transactions.

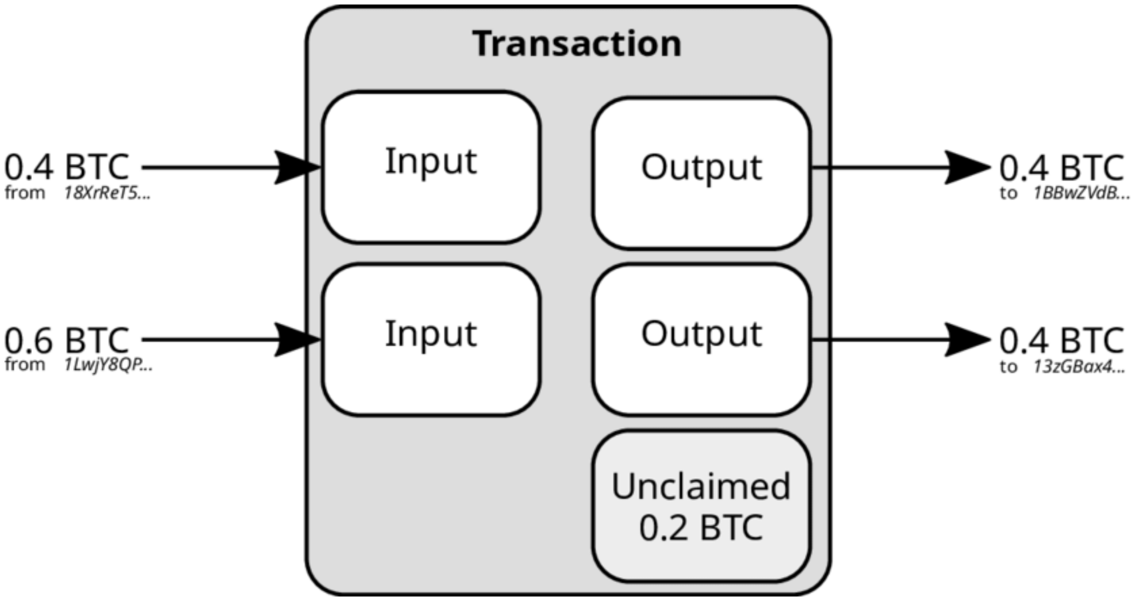


Figure 3.3: Example of Bitcoin transaction with fee for miner as “Unclaimed”. Image source [2].

3.6 Total supply

Blockchains are usually divided into two groups by the way they handle the supply of all coins – one has maximum capped supply as it is the case in Bitcoin and the other group without the maximum supply as in Ethereum.

When a blockchain has a capped supply it will eventually deplete the block rewards. After which point there will be no more generated incentive for miners or validators from the block reward, and they will rely solely on transaction fees.

This resulted in the second type of blockchains without this limitation as a simple form of a workaround. However, this results in constant inflation of new coins into the circulation.

3.7 Transaction fees

A transaction fee is a fee included with a transaction which serves multiple purposes. Some of the main ones are the following:

- It incentivizes the miner or validator to accept our transaction into his mempool. Furthermore, with increasing the transaction fee we can ensure a higher chance of having our transaction included in a block as there is limited size of transactions, which can be included in a block because of a block size.
- It helps to prevent flooding the blockchain with transaction.
- To execute a code in a transaction, most notably a smart contract execution on blockchains such as Ethereum. This can be understood as paying a fee for running the code on the consensus node.

Feerate is a ratio of transaction fee and size of transaction in bytes [23]. From the side of miner or validator this can be seen as how profitable it is to prioritize including this

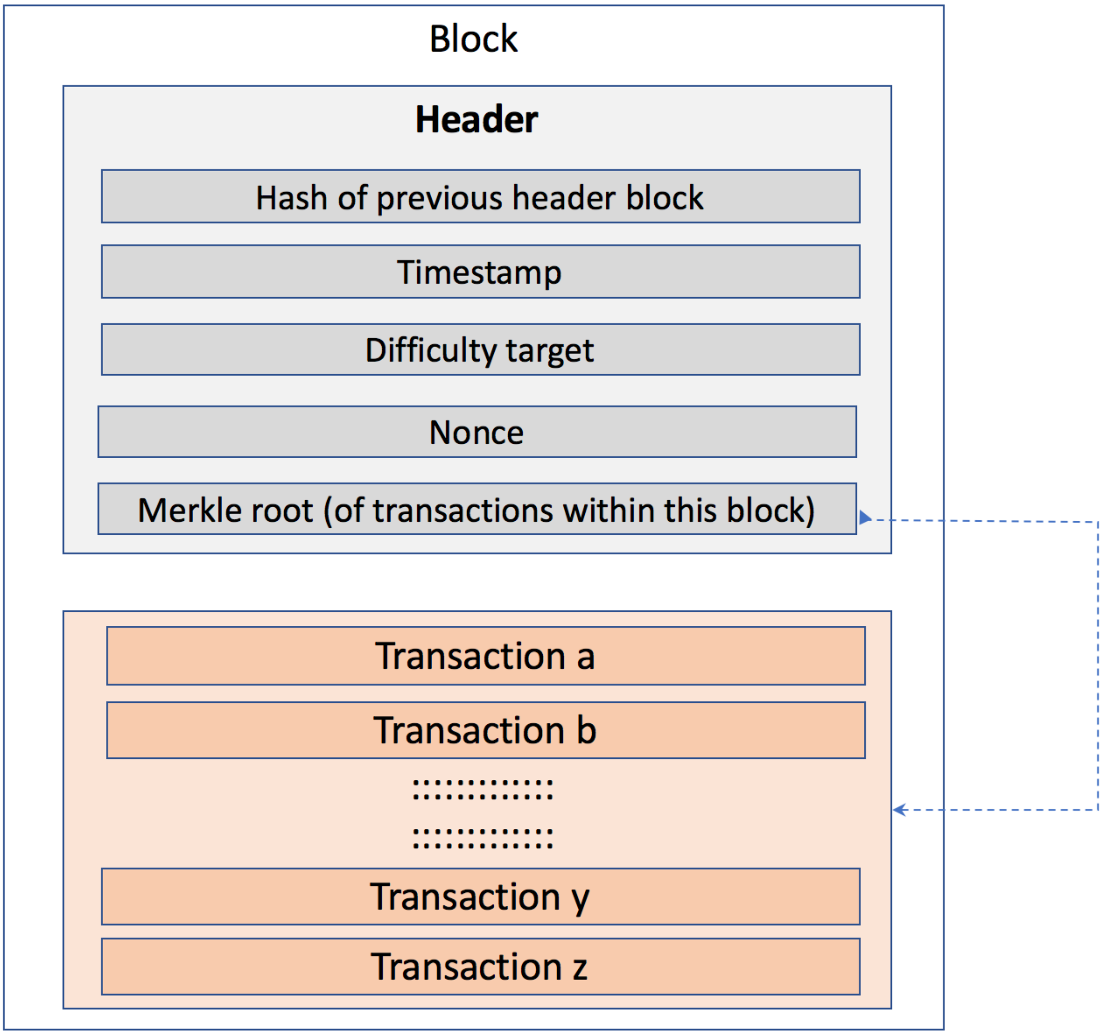


Figure 3.4: Example of Bitcoin block. Image source [3].

transaction over other transactions and miners or validators usually sort transactions in their respective mempool based on this parameter.

Blockchains with capped supply will eventually incentivize miners or validators only with transactions fees.

3.8 Proof of Work

Proof of Work (PoW) blockchains are a type of blockchains, where consensus is reached by dedicating computing power to calculate cryptographic hashes in order to create a new block in the blockchain by the process called mining.

Mining is done by a miner who is seen by the system as a consensus node. This way a miner secures the protocol and in return for finding a block he expects a reward to cover his expenses for mining equipment and utilities.

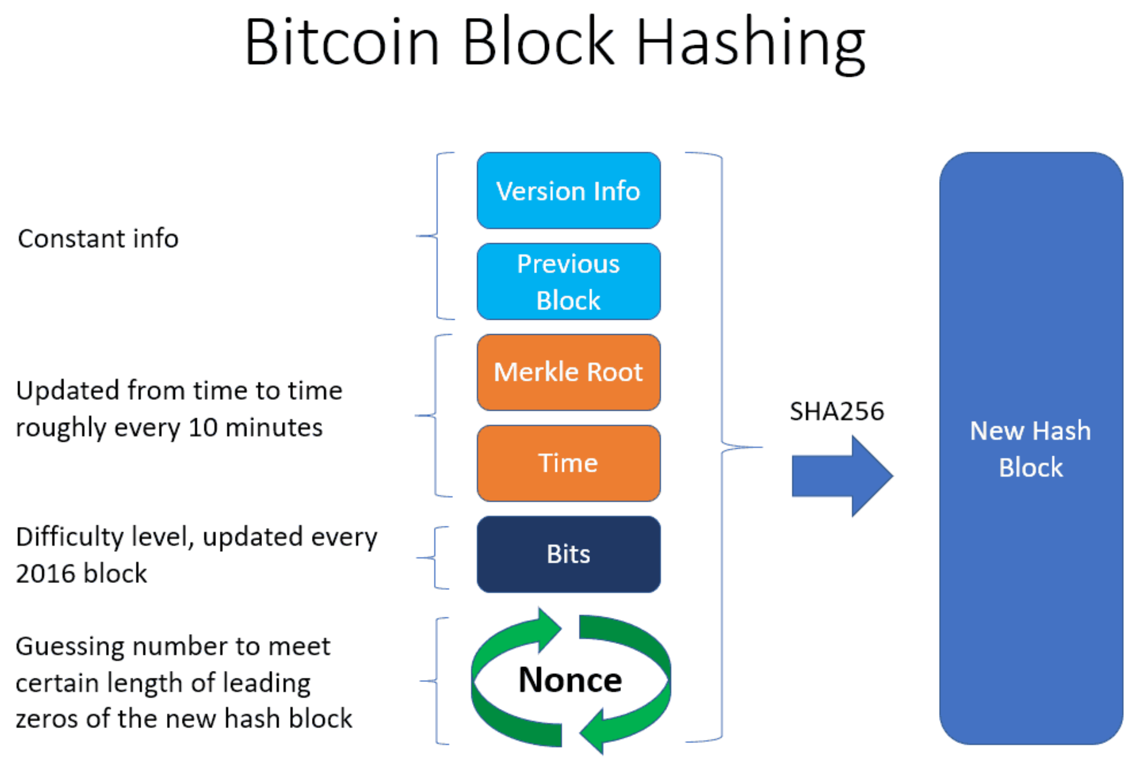

Bitcoin Block Hashing
Constant info
Updated from time to time roughly every 10 minutes
Difficulty level, updated every 2016 block
Guessing number to meet certain length of leading zeros of the new hash block
Version Info
Previous Block
Merkle Root
Time
Bits
Nonce
SHA256
New Hash Block

Figure 3.5: Process of mining in PoW blockchain. Image source [15].

Higher the computing power, called hashrate, a blockchain has, the more secured it can be considered. That is why the system has to make sure it rewards and incentivizes miners to mine in such a system.

3.9 Mining

Is a process specific to PoW, done by a miner. He tries to find such order of block parameters most notable a value of nonce, where the resulting hash of this block's header would be under a target set by the system. With more miners joining the network, the smaller the chance of finding such solution. This leads to miners grouping together and forming mining pools.

Next to the block reward a miner receives in return from mining, he also receives transaction fees from transactions he includes in the final block. If a blockchain does not provide the block reward, a miner relies solely on these transaction fees as his only income from mining and securing the blockchain. This is why he might prefer some transactions over other transactions to maximize his revenue.

Chapter 4

Current Problems with Transaction-Fee-Based Regimes

In this chapter, we will briefly discuss the problems of PoW blockchains relying mostly only on transactions fees to incentivize miners to secure such blockchains. These problems were deeply examined in the original paper from 2016 [5], which this thesis builds upon and provides a solution for few of the presented shortcomings. These shortcomings were displayed on Bitcoin blockchain with additional simulations.

4.1 Fluctuation of miner's reward

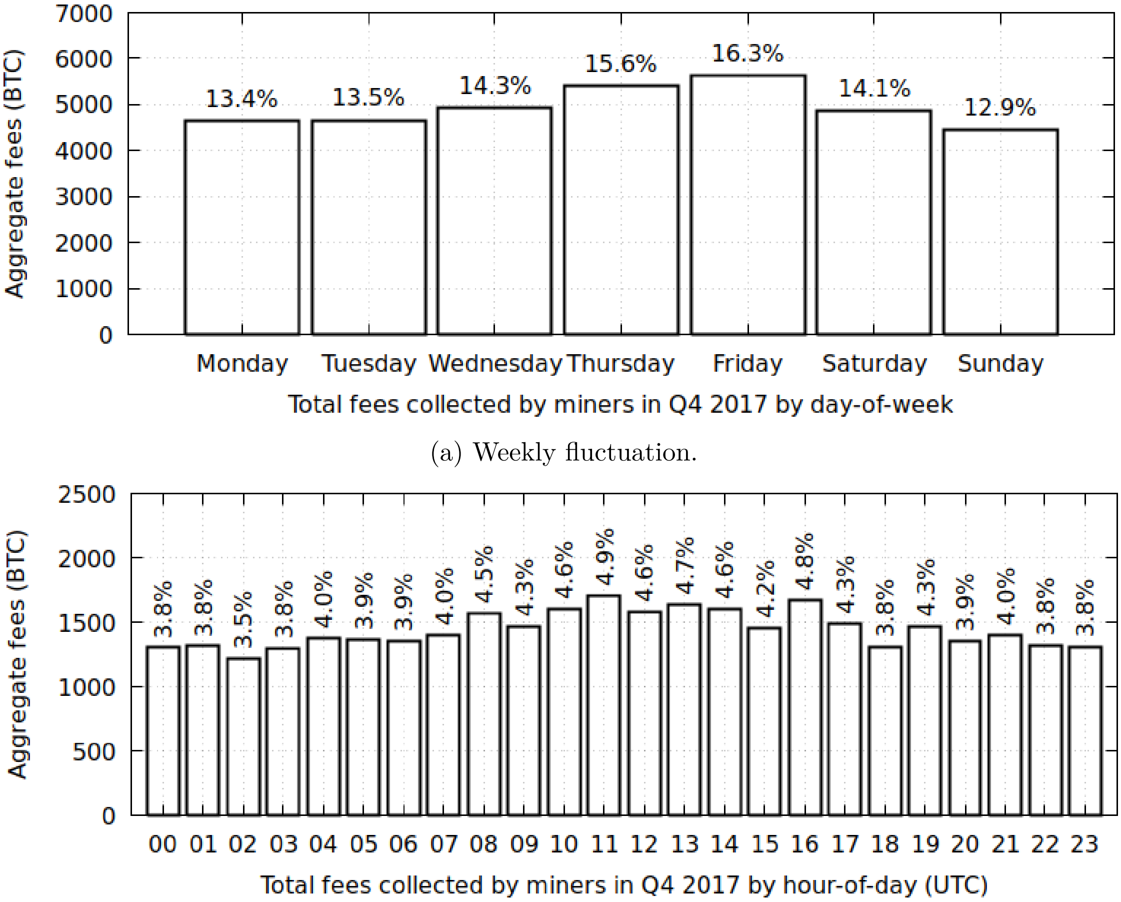
Fluctuations can be split into 2 main categories. Into the first category fall unpredictable fluctuations, which happen from ad hoc events. This can be an example of high volume of transactions in short period of time, such as we have seen for example on blockchain as Ethereum with NFT collection mints. We do not take this category under consideration in this thesis, especially as they are subject of brief time periods. Furthermore, it would be impossible to predict these events on protocol level in advance.

The second group of fluctuations is somewhat predictable and repeatable. Having common averages of volumes in transactions regularly repeated over a presented period of time, which will be subject to when we talk about the fluctuation of miner's reward (revenue). Such fluctuations can be seen as average usage of blockchain via transactions with coins and tokens exchanged for goods and services on a rather repeatable basis.

We can see such example in Figure 4.1 displaying averaged transaction fees during two different time frames collected in Q4 of 2017 on Bitcoin blockchain. We can see the discrepancies might be over 15% from one day to another within a week's time period.

Transaction fees are direct miner's revenue on Bitcoin blockchain in situation, where the block rewards are fully depleted or are insignificant compared to transaction fees. And such fluctuations in transaction fee rewards have some undesirable effects. With the assumption of most miners securing our blockchain as long as they are profitable, high volatility in their revenue might repel them from mining and securing our blockchain. Thus, directly resulting in less secured protocol for such PoW blockchains.

Apart from the example in Figure 4.1 we can also assume similar scenario across year's time period with holidays during different parts of year, e.g. we can assume increased demand before Christmas holidays and decrease after New Year's Eve. We would like to mitigate the effect of these repeatable fluctuations for the sake of all participants to result



(a) Weekly fluctuation.

(b) Daily fluctuation.

Figure 4.1: Fluctuation in transaction fees Q4 2017. Data provided by [23].

in more predictable and less volatile environment for miners securing our protocol and resulting in more secured protocol leading to better user experience.

4.2 Undercutting attack

This attack was firstly introduced and researched in afford-mentioned paper [5]. The original authors ran few simulations proving it a real type of attack.

The malicious miner, who was introduced in [5], attempts to obtain transaction fees by re-mining a top block of the longest chain. By re-mining the top block, he includes a subset of all transactions in his newly re-mined block. The rest of these “generous” transactions (having higher feerate Paragraph 3.7 than transactions from surrounding blocks) from the originally mined block, is left in the mempool. This is done by not being included in the re-mined block, leaving them unclaimed for the next miner. This is showed in Figure 4.2.

By leaving some transactions from the original *undercut* block the malicious miner motivates the next miner to mine on top of his block, yielding such miner additional revenue by allowing him to include these unclaimed transactions from the mempool. By miners mining on top of this re-mined block they take part in the undercutting attack.

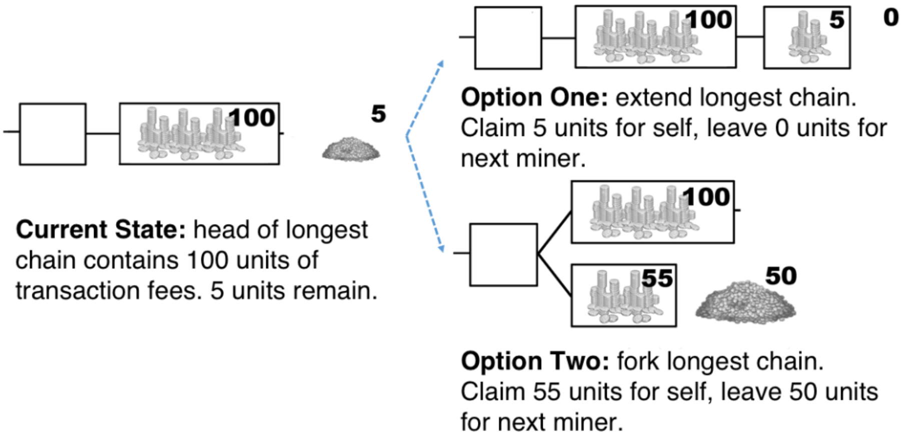

100
5
100
5
0
Option One: extend longest chain. Claim 5 units for self, leave 0 units for next miner.
Current State: head of longest chain contains 100 units of transaction fees. 5 units remain.
100
55
50
Option Two: fork longest chain. Claim 55 units for self, leave 50 units for next miner.

Figure 4.2: Undercutting attack from the original paper [23].

This is especially valid strategy for certain blocks. Where such block has much higher value compared to blocks before it, meaning this block yielded its miner significantly higher revenue, than to miners of few previous blocks. Furthermore, if this block takes most of "generous" transactions from the mempool, leaving only transactions with smaller feerate. Thus, such block can be a subject of undercutting attack. To provide a simple example we can again consider a new NFT series drop and block minting this series would have a much higher feerate for minting transactions in order to ensure the drop.

Also it is assumed the blockchain contains miners, who are honest except they do not obey the longest-chain rule but rather try to prioritize their own profitability. This results in leaving the true honest miners, following the longest-chain rule, into a disadvantage. Successful execution of undercutting attacks might directly result in higher orphan rate[1], unreliability of the system (such as longer block time) and double spending.

4.3 Mining gap

If the protocol does not provide any block reward and all the miner's revenue comes from the transaction fees from transactions, he includes in a block, the miner no longer has a guarantee of being profitable during the whole time he mines even if he successfully mines a block. This is the main premise behind the mining gap.

This comes from the fact, that when a new block is mined and added to the blockchain and a miner receives this update, he would have to remove all the included transactions in this block from his mempool. Leaving the miner less transactions in the mempool he can include in the next block. Thus, it is argued a miner does not have to be able to covert his expenses such as electricity costs, internet connection costs etc.

We will describe the situation of miner mining right after a new block was found. He would have to remove all transactions included in this block from his mempool thus leaving him possibly no transactions. However, he would still be trying to mine for the next block, increasing his expenses in the meanwhile, for potentially finding a block with zero transactions. Even if he found such block, he would not gain any profit from it as it

[1] ratio of valid blocks which are not part of the chain. This also happens normally because of network delay etc.

might have few to none fees from transactions. Such situation persists even as the miner is receiving new transactions, as they might not be coming fast enough in the beginning.

Not mining and waiting for the mempool to fill with new transactions to start mining again might not be a feasible strategy. For ASIC-resistant blockchains, a miner might decide to hop between such blockchains, as he is not restricted to mining on PoW blockchains secured with the same cryptographic primitives. However, this is rather speculative guess.

4.4 Extisting solutions

The work of Carlsten et al. [5] is the inspiration for our paper, which for the first time describes undercutting attacks arising from the exponential distribution of block creation time and significant differences in transaction fees. The authors simulated Bitcoin under transaction-fee-based regime and found that there exist the minimal threshold of DEFAULT-COMPLIANT miners equal to 66%. However, our paper is not the first trying to target some subset of problems discovered or mentioned in afford-mentioned work. We will take a look at a few similar solutions. Furthermore, we would like to extend some knowledge displayed in these works and the facts we have used throughout our own work.

Gong et al. [10] argue that using all accumulated fees in the mempool regardless of the block size limit is infeasible in practice and can inflate the profitability of undercutting that was originally described in [5]. Furthermore, Houy [12] demonstrates that a constraint on the block size limit (thus the number of transactions) has economic importance and allows transaction fees not dropping to zero. Therefore, Gong et al. [10] model the profitability of undercutting with the block size limit presented, which bounds the claimable fees in a mining round. The authors presented a countermeasure that selectively assembles transactions into the new block, while claiming fewer fees to avoid undercutting. We argue that in contrast to our approach, this solution cannot be enforced by the consensus protocol, and thus might still enable undercutting to occur. It tries to preemptively include only some subset of transactions with high feerate and leave some others for the next miner, resulting in rather preventive solution.

However, based on facts from these works we have decided with updating the simulator we have included changes simulating “fullMempool” option Paragraph 6.1.4, which essentially results in having capped amount of fees per simulated block. This fact can be also usually confirmed on the Bitcoin blockchain using [17], where we during the creation of our work saw multiple blocks being filled with unconfirmed transactions.

Zhou et al. [24] deal with the problem of a mining gap, which is more significant when the throughput of blockchain is high. Therefore, the authors propose the self-adaptive algorithm to adjust the block size every 1000 blocks and thus ensure that blocks have enough space to pack new transactions.

Lastly, even though Bitcoin-NG paper [9] does not present offer a solution for our problems, which we focus on in this paper, we found a similar idea of splitting fees as we introduce with our ratio $\mathbb{C}$ and $\mathbb{M}$ respectively in Chapter 5. The subject of Bitcoin-NG [9] is to propose a new consensus mechanism, however it also contains an idea of splitting the transaction fees between two entities – the current leader and the miner of the block – which should incentivize the miner to include blocks created by the leader. This is similar approach in a way, as we also allow the miner to take only portion of the fees directly.

However, the other part goes to a different party and no averaging is happening here. Furthermore, the main point of paper [9] is to showcase a blockchain with some features of Bitcoin while being more scalable with smaller confirmation time.

Chapter 5

Proposed Solution

Here, we describe our proposed solution in more detail. In sum, our proposed solution collects a percentage from all transaction fees, which are usually paid in a native cryptocurrency coin, collected in the mined blocks into one or multiple fee-redistribution contracts (i.e., $\mathcal{FRC}$s). Miners of the blocks, who must contribute to these contracts, are at the same time rewarded from them, while the received reward approximates a moving average of the incoming transaction fees across the fixed sliding window of the blocks.

The fraction of transaction fees (i.e., $\mathbb{C} \in \langle 0, 1 \rangle$) from the mined block is sent to $\mathcal{FRC}$s and the remaining fraction of transaction fees (i.e., $\mathbb{M} \in \langle 0, 1 \rangle$) is directly assigned to the miner, such that $\mathbb{C} + \mathbb{M} = 1$.

The role of $\mathbb{M}$ is to keep the incentive for the miners to prioritize transactions with higher feerate Paragraph 3.7 in tack, in order to keep the free-market biding feature unchanged. Leaving this feature for users creating transactions as option to have a chance to bid higher feerate in order to have their transaction included in sooner block.

While the role of $\mathbb{C}$ is to mitigate the problems, this paper is focused on, such as fluctuating miner's revenue, undercutting attacks and the mining gap by averaging the collected fees. And rewarding the miner with averaged reward over different time period, based on individual contracts. Roughly said, a miner can expect $\mathbb{M}$ of his reward directly from the collected fees and $\mathbb{C}$ as averaged reward of fees collected in individual contracts.

5.1 Overview

We depict the overview of our approach in Figure 5.1, and it consists of the following steps:

1. Using $\mathcal{FRC}$, the miner calculates the reward for the next block B he receives from the (i.e., $nextClaim(\mathcal{FRC})$ – see Equation 5.4) that will be paid by $\mathcal{FRC}$ to the miner of that block.

2. The miner mines the block B using the selected set of transactions with the highest feerate to maximize the profit from his mempool.

3. The miner of the mined block B directly receives a certain fraction of all the collected transaction fees (i.e., $B.fees\ *\ \mathbb{M}$) and the remaining part (i.e., $B.fees\ *\ \mathbb{C}$) the miner must send to $\mathcal{FRC}$.

4. The miner obtains $nextClaim$ from $\mathcal{FRC}$.

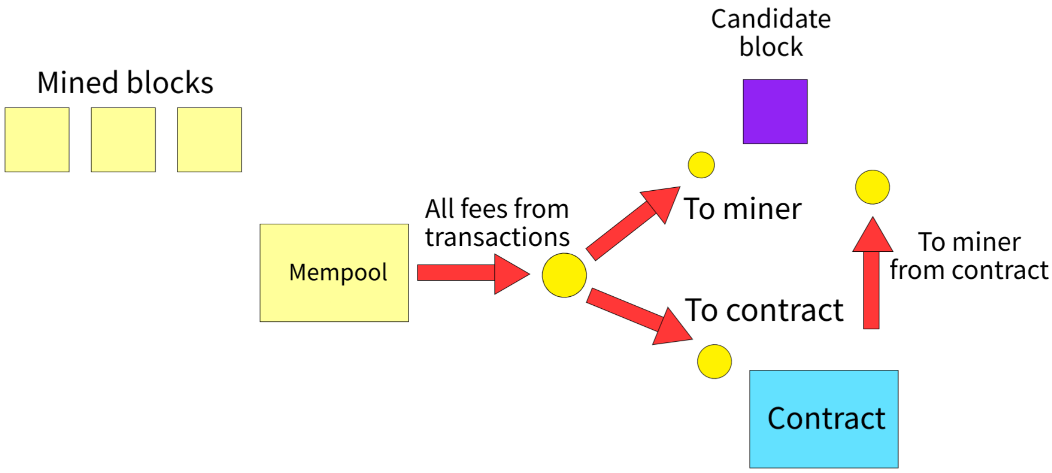


Figure 5.1: Overview of our solution.

Our approach is embedded into the consensus protocol, and therefore consensus nodes are obliged to respect it in order to ensure that their blocks are valid, accepted by the network, resulting in receiving rewards from mining such blocks. It can be implemented utilizing smart contracts [20] of the underlying blockchain platform Appendix A. In the environment with constant transaction fees, a miner would receive the same amount with or without our solution.

5.2 Prioritization of transactions with higher feerate

In public blockchains (especially with the transaction-fee-based regime) there exists a mechanism to ensure prioritization in processing of transactions with higher feerate, which might result into fluctuations in rewards of the miners. This feature allows users to have their transactions processed in a sooner block if the mempool holds too many transactions to be fit into a single block. In our approach, we preserve the transaction prioritization since we directly attribute a part of the transaction fees to the miner (i.e., $\mathbb{M}$).

5.3 Fee-Redistribution Contracts

We define the fee-redistribution contract as a following tuple:

$$\mathcal{FRC} = (\nu, \lambda, \rho), \tag{5.1}$$

where

- ν is the accumulated amount of usually native blockchain coins in the contract &
- λ denotes the size of $\mathcal{FRC}$' sliding window in terms of the number of preceding blocks that contributed to ν &

- And ρ is the parameter defining the ratio for redistribution of incoming collected transaction fees to the particular $\mathcal{FRC}$ among multiple contracts, while the sum of ρ across all $\mathcal{FRC}$s must be equal to 1 (i.e. see Equation 5.2).

$$\sum_{x \ \in \ \mathcal{FRC}s} x.\rho \quad = \quad 1. \tag{5.2}$$

In contrast to a single $\mathcal{FRC}$, multiple $\mathcal{FRC}$s enable better adjustment of compensation to miners during periods of higher transaction fee fluctuations or in an unpredictable environment (we show this in Section 6.4). With multiple $\mathcal{FRC}$s we can aim individual $\mathcal{FRC}$ to target different time periods as mentioned in Section 4.1. We denote the state of $\mathcal{FRC}$s at the blockchain's height H as $\mathcal{FRC}s_{[H]}$. Then, we determine the reward from $\mathcal{FRC}_{[H]} \in \mathcal{FRC}s_{[H]}$ for the miner of the next block with height $H+1$ as follows:

$$\partial Claim_{[H+1]}^{\mathcal{FRC}_{[H]}} = \frac{\mathcal{FRC}_{[H]}.\nu}{\mathcal{FRC}_{[H]}.\lambda}, \tag{5.3}$$

while the reward obtained from all $\mathcal{FRC}$s is

$$nextClaim_{[H+1]} = \sum_{\mathcal{X}_{[H]} \ \in \ \mathcal{FRC}s_{[H]}} \partial Claim_{[H+1]}^{\mathcal{X}_{[H]}}. \tag{5.4}$$

Then, the total reward of the miner who mined the block $B_{[H+1]}$ with all transaction fees $B_{[H+1]}.fees$ is

$$rewardT_{[H+1]} = nextClaim_{[H+1]} + \mathbb{M} * B_{[H+1]}.fees. \tag{5.5}$$

The new state of contracts at the height $H+1$ is

$$\mathcal{FRC}s_{[H+1]} = \{\mathcal{X}_{[H+1]}(\nu, \lambda, \rho) \ \ | \tag{5.6}$$

$$\begin{aligned}
\lambda &= \mathcal{X}_{[H]}.\lambda, && (5.7)\\
\rho &= \mathcal{X}_{[H]}.\rho, && (5.8)\\
\nu &= \mathcal{X}_{[H]}.\nu - \partial Claim_{[H+1]} + deposit * \rho, && (5.9)\\
deposit &= B_{[H+1]}.fees * \mathbb{C}\}, && (5.10)
\end{aligned}$$

where *deposit* represents the fraction $\mathbb{C}$ of all collected transaction fees from the block $B_{[H+1]}$ that are deposited across all $\mathcal{FRC}$s in ratios respecting Equation 5.2.

5.4 Example

We present an example using Bitcoin [18] to demonstrate our approach. We assume that the current height of the blockchain is H, and we utilize only a single $\mathcal{FRC}$ with the following parameters:

$$\mathcal{FRC}_{[H]} = (2016, 2016, 1).$$

We set $\mathbb{M} = 0.4$ and $\mathbb{C} = 0.6$, which means a miner directly obtains 40% of the $B_{[H+1]}.fees$ and $\mathcal{FRC}$ obtains remaining 60%.

Next, we compute the reward from $\mathcal{FRC}$ obtained by the miner of the block with height $H+1$ as

$$\partial Claim_{[H+1]} = \frac{\mathcal{FRC}_{[H]}.\nu}{\mathcal{FRC}_{[H]}.\lambda} = \frac{2016}{2016} = 1 \text{ BTC},$$

resulting into

$$nextClaim_{[H+1]} = \partial Claim_{[H+1]} \ = \ 1 \text{ BTC}.$$

Further, we assume that the total reward collected from transactions in the block with height $H+1$ is $B_{[H+1]}.fees = 2$ BTC. Hence, the total reward obtained by the miner of the block $B_{[H+1]}$ is

$$\begin{aligned} rewardT_{[H+1]} &= nextClaim_{[H+1]} + \mathbb{M} * B_{[H+1]}.fees \\ &= 1 + 0.4 * 2 \\ &= 1.8 \text{ BTC}, \end{aligned}$$

and the contribution of transaction fees from $B_{[H+1]}$ to the $\mathcal{FRC}$ is

$$deposit = B_{[H+1]}.fees * \mathbb{C} = 1.2 \text{ BTC}.$$

Therefore, the value of ν in $\mathcal{FRC}$ is updated at height H + 1 as follows:

$$\begin{aligned} v_{[H+1]} &= \mathcal{FRC}_{[H]}.\nu - nextClaim_{[H+1]} + deposit \\ &= 2016 - 1 + 1.2 \text{ BTC} \\ &= 2016.2 \text{ BTC}. \end{aligned}$$

5.5 Traditional way in transaction-fee-based regime

In traditional systems (running in transaction-fee-based regime) $rewardT_{[H+1]}$ would be equal to the sum of all collected transaction fees $B_{[H+1]}.fees$ (i.e., 2 BTC); hence, using $\mathbb{M} = 1$. In our approach, $rewardT_{[H+1]}$ can only be equal to the sum of all transaction fees in the block $B_{[H+1]}$, if:

$$B_{[H+1]}.fees = \frac{nextClaim_{[H+1]}}{\mathbb{C}}. \tag{5.11}$$

In our example, a miner can mine the block $B_{[H+1]}$ while obtaining the same total reward as the sum of all transaction fees in the block if the transactions carry 1.66 BTC in fees:

$$B_{[H+1]}.fees = \frac{1}{0.6} = 1.66 \text{ BTC}.$$

Even though at first sight this might appear as defect it is by design and in Section 6.5 we will show how our design makes mining on blockchains with only transaction fees more predictable and sustainable.

5.6 Initial setup of $\mathcal{FRC}$s contracts

We propose to initiate the contracts in our approach by some genesis value that enables even start or implement this feature on existing blockchains such as Bitcoin while it is generating significantly higher miner's revenue from block reward compared to transaction fees.

To enable an even start, we propose to initiate $\mathcal{FRC}$s of our approach by a genesis value. The following formula calculates the genesis values per $\mathcal{FRC}$ and initializes starting state of $\mathcal{FRC}s_{[0]}$:

$$\{\mathcal{FRC}^{x}_{[0]}(\nu, \lambda, \rho) \mid \nu = \overline{fees} * \mathbb{C} * \rho * \lambda\}, \tag{5.12}$$

where $\overline{fees}$ is the expected average of transaction fees coming from users of the blockchain.

Note, this paper is not subject to setting individual parameters in the final blockchain. Rather, it tries to demonstrate how this solution works and how different parameters influence the behavior. The parameters should be adjusted on individual basis for blockchains and should be considered by the creators to match their expected features.

Chapter 6

Experiments

In this chapter, we will focus on conducting multiple experiments in order to explore a way to evaluate our solution. To showcase how different parameters influence the behavior of our solution and what for should individual parameters be used for. We will provide a summary overview of all important parameters in the beginning of each experiment.

6.1 Simulator

6.1.1 General overview

The simulator is of type Discrete system, and its behavior can be described in 5 steps, however it still follows the simple 3 steps of initialization of environment, execution of simulation and lastly harvesting and representing the results.

During initialization the simulator calculates a target time period, meaning it calculates for how many simulation seconds it will run. This is calculated from input parameter of expected blocks in the final blockchain. This parameter gets multiplied by the expected blocktime and results in defining the simulator's timeline. Furthermore, it sets up settings for blockchain characteristic features as expected blocktime, block reward, incoming fees etc., and also initializes individual learning miners (our learning models) to initial values and defined spread of miners. Original authors [5] have included 2 main learning strategies, multiplicative and EXP3.

Now the simulator proceeds to execute individual runs of simulation, which involves steps from 2nd to 4th of this simulator. First step is to reset the blockchain to default state and prepare our learning models for miners before the run of a single simulation. The next step is to simulate this run, which is described later in Section 6.1.2. Lastly we evaluate the resulting blockchain and profitability of individual miners (in case we have multiple mining strategies) and update weights of all our mining strategies.

At the final stage of the simulation, the simulator prints summary information and leaves files with the final weights of the different mining strategies. With our changes, the simulator can also leave files with the information from the last run of simulation about the final blockchain's block rewards collected by the miners of corresponding blocks and the fee scenario for that run of simulation.

6.1.2 Single run of simulation

In each event, either an advancement in time by a second or new block found, we let individual miners, with different strategies, calculate their chances of finding a block. During the calculation of chances of finding a block, the miner considers more information for his next steps from the blockchain available to him. These might include the value of top block, transactions in the mempool etc., which can be used by his strategy. This is also true for other parts of simulation mentioned below. However, in order to keep this overview simple we omit these details.

Then a broadcast round takes place, where depending on the length of miners' private chain and his strategy, he decides whether to broadcast his chain or not. Here, the simulator allows using network delays, which can be optionally set up in the initialization phase. Note, the honest miner broadcasts and publishes as soon as he finds a block. Up next, a publishing round is evaluated from previously broadcasted blocks. After these steps, the known blockchain of all miners should be updated to let them decide on their next mining steps.

These simplified steps are repeated until we reach our target time period.

For more information about the simulator, we advise the reader to read the original paper [5], or read the updated source code (attached with this work), which can be also found on [4].

6.1.3 Changes in simulator

We used the original simulator [14] with slight modifications to the way rewards are calculated and issued to miners from the blocks. We have added an option to create a custom scenario for incoming fees from transactions. This option replaces the fixed income per time unit (second) i.e. linearly incoming total value in fees, however this is still possible with our option. Finally, we have integrated our proposed solution in order to run simulation with our solution and to be able to directly recreate the original experiment for comparison with the problem defined in Section 4.2.

6.1.4 Custom fee scenario

We have added the option to create a configuration JSON file named `feeSimulation.json`. The configuration defines the following properties:

- `mean` – default value of *mean* for normal distribution function
- `deviation` – default value of *deviation* for normal distribution function
- `fullMempool` – boolean value (explained in Paragraph 6.1.4)
- `timeline` – array of individual fee epochs.

Parameters for individual fee epochs

- `start` – blockheight, at which this epoch starts
- `epochType` – specifies strategy for generating fees
- `values` – array expecting integral values used in `epochType`.

And finally individual `epochType`-s

- 0 – fixed reward per block specified by `values[0]`
- 1 – linearly changing fees per block starting from `values[0]` (block `start`) until `values[1]`
- 2 – normal distribution N∼(`values[0]`,`values[1]`),

where the epochs are sorted by the `start` attribute and if the simulation has no more defined epochs, then `epochType` = 2 with values `mean` & `deviation` is used.

The `fullMempool` parameter was introduced in order to allow compatibility with the original simulator but to also allow us to simulate a more realistic behavior such as reproduction of Figure 4.1.

- *True* means mined block gets the amount specified by scenario unrelated to time it took to mine this block.
- *False* results adding $\frac{1}{BLOCKTIME} * seconds$, where
  - $BLOCKTIME$ represents expected blocktime, which is 600 seconds as per Bitcoin
  - $seconds$ represents the time it took to mine this block,

  meaning the longer it takes a miner to mine this block, the higher reward he gets i.e. if he mined this block for average blocktime, he would receive reward specified by the scenario.

We have changed the source code to reflect the expected behavior, written in the `feeSimulation.json` file. This means how fees are being generated (added) to the simulated protocol during simulation to not break the expected functionality of the simulator.

6.1.5 Contracts

Similarly to the previous point, we have created a JSON config file named `feeContracts.json` to set up the $\mathcal{FRC}$s in the simulator, where individual $\mathcal{FRC}$ corresponds to the definition in Equation 5.1. The following parameters are expected in the config file:

- `toContracts` – corresponds to $\mathbb{C}$
- `contracts` – array of individual contracts,

where individual contracts are defined with following parameters:

- `value` – corresponds to $\mathcal{FRC}.\nu$
- `length` – corresponds to $\mathcal{FRC}.\lambda$
- `percentage` – corresponds to $\mathcal{FRC}.\rho$.

Furthermore, we have adjusted how a miner calculates his expected reward for individual mining strategies as this influences how a miner decides, on which chain he mines or whether he will execute the undercutting attack. These strategies are defined in the simulator, and they simulate differently behaving miners such as honest miner, undercutting miner etc. This had to be done as with the introduction of $\mathcal{FRC}$s a miner is not able to undercut the whole value of a block but just a $\mathbb{M}$ fraction from the total block value.

6.1.6 Input arguments

In order to make running simulation easier and faster without the need to recompile the whole executable with every slight modification, except the two JSON config files previously mentioned, we have decided to pass some commonly used parameters to the executable. These parameters are following in their respective order:

- expected number of blocks in the longest chain in a single run of simulation, not considering forks
- total number of simulation runs.

6.1.7 Common features of experiments

We evaluated our proof-of-concept implementation of $\mathcal{FRC}$s on a long term scenario that we designed to demonstrate significant changes in the total transaction fees in the mempool evolving across the time. This scenario is depicted in the resulting graphs of most of our experiments, where our designed scenario of fees is represented by the "*Fees in mempool*" series such as Section 6.2 and Section 6.3.

We experimented with different parameters and investigated how they influenced the total rewards miners receive from $\mathcal{FRC}$s versus the baseline without our solution. Mainly these included a setting of $\mathbb{C}$ as well as different lengths of individual λs of $\mathcal{FRC}$s. Note that we used the value of transaction fees per block equal to 50 BTC, alike in the original paper introducing undercutting attacks [5]. This corresponds to the $\overline{fees} = 50BTC$, accordingly to which we created the initial setup of $\mathcal{FRC}$s as per Equation 5.12, and usually the start of the designed scenario starts from this value in terms of fees.

Also, in all our experiments but the last one (i.e., Section 6.6), we enabled the fullMempool option Paragraph 6.1.4 (i.e. `True`, meaning fixed reward per block) to ensure more realistic conditions. Besides this option we also used only honest miners across all our experiments except the last one, which is based around different, some of them malicious, miners.

Lastly, many experiments are conducted under less usual conditions, compared to reality, to allow us to better understand the impact of our solution.

6.2 Experiment I – Different lengths of $\mathcal{FRC}$s

We will observe both parameters simultaneously (i.e. ratio of $\mathbb{M}\&\mathbb{C}$ and $\mathcal{FRC}.\lambda$) as separating them is not a viable option.

6.2.1 Methodology

The purpose of this experiment was to investigate the amount of the reward a miner received with our approach versus the baseline (i.e., the full reward is based on all transaction fees). In this experiment, we investigated how $\mathbb{C}$ influences the total reward of the miner and how λ of the sliding window averaged the rewards. In detail, we created two independent $\mathcal{FRC}$s with different λ – one was set to 2016 (i.e., $\mathcal{FRC}^1$), and the second one was set to 5600 (i.e., $\mathcal{FRC}^2$). We simulated these two $\mathcal{FRC}$s with three different values of $\mathbb{C} \in \{0.5,\ 0.7,\ 0.9\}$ while keeping the same scenario. The defined $\mathcal{FRC}^1$ and $\mathcal{FRC}^2$ respectively:

$$\mathcal{FRC}s^1 = \{\mathcal{FRC}_1(\_, 2016, 1)\}$$
$$\mathcal{FRC}s^2 = \{\mathcal{FRC}_1(\_, 5600, 1)\}$$

Note, the *value* is dynamically calculated based on value of $\mathbb{C}$ using Equation 5.12 with $\overline{fees} = 50BTC$ for the initial setup.

6.2.2 Results

The results of this experiment, are depicted in Figure 6.1. Across all runs of our experiment we can observe that $\mathcal{FRC}^2$ adapts slower as compared to $\mathcal{FRC}^1$, which leads to more significant averaging of the total reward paid to the miner. This means with falling fees, the miners can keep relatively higher compensations. Even though more significant averaging is desired, we can see faster adaptation to changing environment by $\mathcal{FRC}^1$, what a miner would prefer with rising fees.

We can see the averaging of the final reward is at the same time influenced by ratio of $\mathbb{M}\&\mathbb{C}$. The higher the $\mathbb{C}$ parameter, the higher the averaging. However, the less direct reward to the miner from the transaction fees he includes in a block. Higher $\mathbb{C}$ is desired, however it is questionable at what point we will see diminishing returns in terms of miners not needing to prioritize higher feerate transactions.

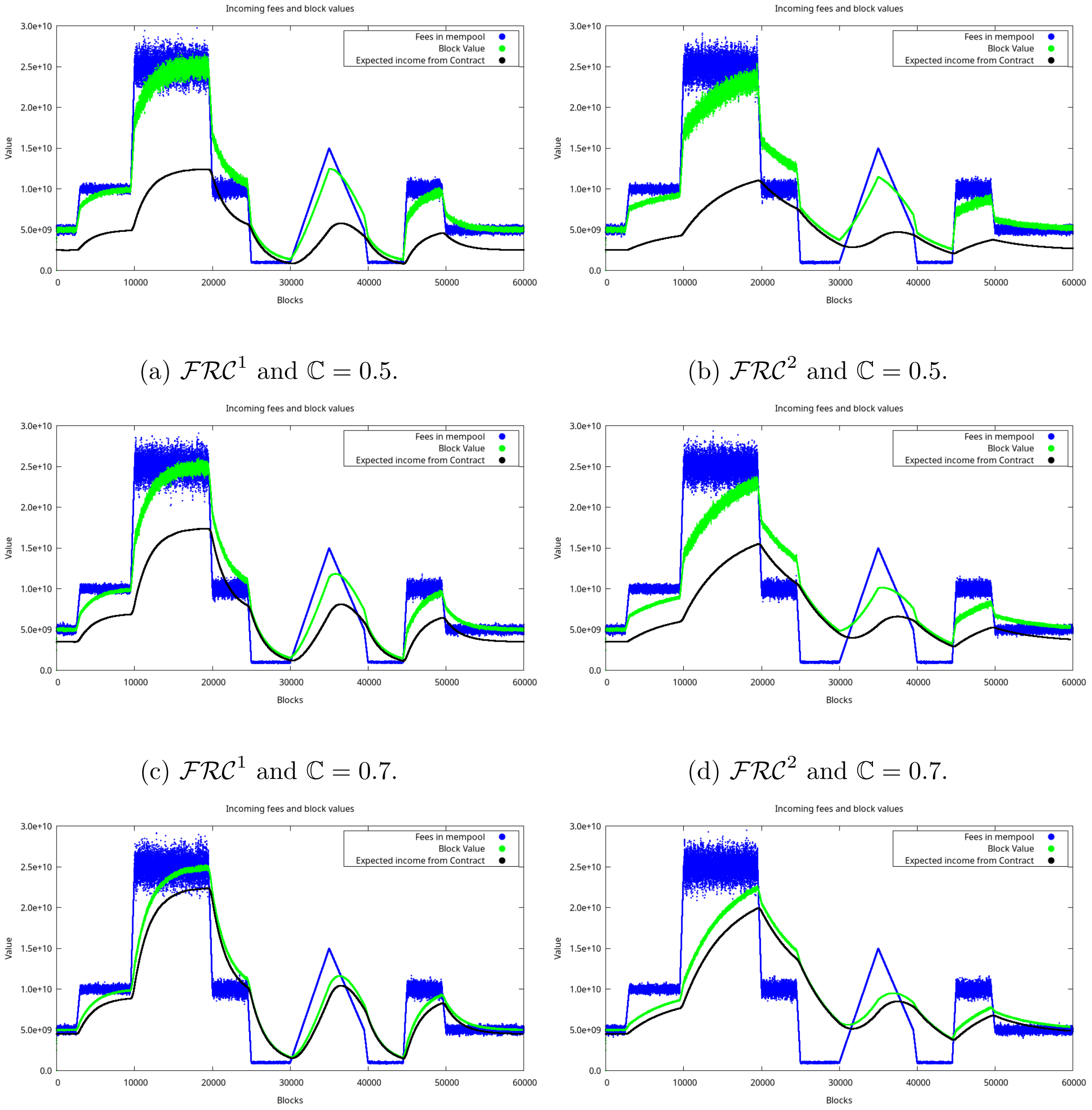
6.3 Experiment II – Multi-contract $\mathcal{FRC}$

6.3.1 Methodology

In this experiment, we investigated how multiple contract $\mathcal{FRC}$s dealt with the same scenario as before – i.e., varying $\mathbb{C}$. Furthermore, we investigated in detail how individual $\mathcal{FRC}$s contributed to the $nextClaim_{[H+1]}$ by their individual $\partial Claim_{[H+1]}^{\mathcal{FRC}_{[H]}}$. This time, we varied only the parameter $\mathbb{C} \in \{0.5,\ 0.7,\ 0.9\}$, and we considered four $\mathcal{FRC}$s setup:

$$\begin{gathered}\mathcal{FRC}s = \{\\ \mathcal{FRC}_1(\_, 1008, 0.07), \mathcal{FRC}_2(\_, 2016, 0.14),\\ \mathcal{FRC}_3(\_, 4032, 0.28), \mathcal{FRC}_4(\_, 8064, 0.51)\},\end{gathered}$$

where their lengths λ were set to consecutive powers of two (to see differences in more intensive averaging spread across longer intervals), and their redistribution ratios ρ were set



(a) $\mathcal{FRC}^1$ and $\mathbb{C} = 0.5$.

(b) $\mathcal{FRC}^2$ and $\mathbb{C} = 0.5$.

(c) $\mathcal{FRC}^1$ and $\mathbb{C} = 0.7$.

(d) $\mathcal{FRC}^2$ and $\mathbb{C} = 0.7$.

(e) $\mathcal{FRC}^1$ and $\mathbb{C} = 0.9$.

(f) $\mathcal{FRC}^2$ and $\mathbb{C} = 0.9$.

Figure 6.1: Experiment I investigating various $\mathbb{C}s$ and λs of a single $\mathcal{FRC}$, where $\mathcal{FRC}^1.\lambda = 2016$ and $\mathcal{FRC}^2.\lambda = 5600$. *Fees in mempool* show the total value of fees in the mempool able to be mined (i.e., representing the baseline). *Block Value* is the actual reward a miner received in block B as a sum of the fees he obtained directly (i.e. $\mathbb{M} * B.fees$) and the reward he got from $\mathcal{FRC}$ (i.e., $nextClaim_{[H]}$). *Expected income from Contract* represents the reward of a miner obtained from $\mathcal{FRC}$ (i.e., $nextClaim_{[H]}$).

to maximize the potential of averaging by longer $\mathcal{FRC}$s (see details below in Section 6.3.3). However, we will show additional data, that is valuable in multi-$\mathcal{FRC}$ environment. The purpose of this experiment is to show, how different $\mathcal{FRC}$s inside $\mathcal{FRC}s$ setup are affected and their impact on the final reward going to the miner. Meaning, individual $\mathcal{FRC}$s can target different time averaging windows.

6.3.2 Results

The results of this experiment are depicted in Figure 6.2. We have verified the expected advantages of using multi-$\mathcal{FRC}$ setup. We can observe that the shorter $\mathcal{FRC}$s quickly adapted to new changes, and the longer $\mathcal{FRC}$s kept more steady income for the miner. In this sense, we can see that $\partial Claim^4$ held steadily over longer periods of lower fees in the scenario, while for example $\partial Claim^1$ fluctuated more. While taking the advantage of having multiple $\mathcal{FRC}$s we are targeting different time frames with different λs .

Since the scenarios of fees evolution in the mempool was the same across our first two experiments, we can compare the $\mathcal{FRC}$ with $\lambda = 5600$ from Section 6.2 and the current setup involving four $\mathcal{FRC}$s – both had some similarities. This was done intentionally to compare, if we are able to interchange multi-$\mathcal{FRC}$ setup with a single $\mathcal{FRC}$ using $effective\_\lambda$, which we will introduce in Section 6.4.

6.3.3 Different ρs

In Figure 6.3 we investigated different values of ρ in the same set of four contracts and their impact on $\partial Claim$s to ensure our assumptions are correct. The results show how different values of ρs behave in respect to their relative λs of multiple $\mathcal{FRC}$s. To maximize the potential of averaging, the results incline towards a correlation between ρ, and it's respective λ should take place.

6.4 Experiment III – Effective length of $\mathcal{FRC}$

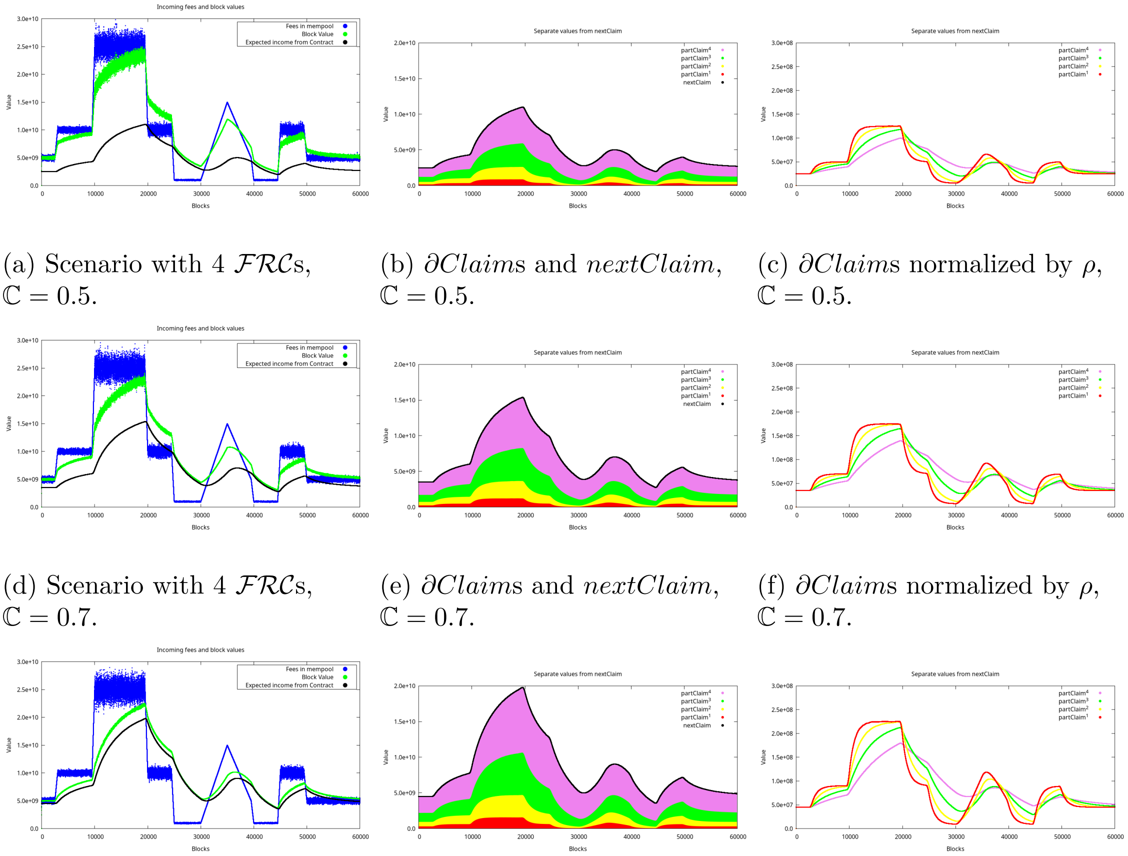
6.4.1 Methodology

In this experiment, we investigated whether it is possible to use a single $\mathcal{FRC}$ setup to replace a multiple $\mathcal{FRC}$s while preserving the same effect on the *nextClaim*. This came as an idea to investigate this option in order to simplify the setup and consolidate multiple $\mathcal{FRC}$s into a single one. This would also result in saving more space in the blockchain. To quantify a difference between such cases, we introduced a new metric of $\mathcal{FRC}s$, called effective_λ, which can be calculated as follows:

$$\text{effective\_}\lambda(\mathcal{FRC}s) = \sum_{x \,\in\, \mathcal{FRC}s} x.\rho * x.\lambda. \tag{6.1}$$

As the example, we were interested in comparing a single $\mathcal{FRC}$ with 4 $\mathcal{FRC}$s, both configurations having the equal effective_λ. The configurations of these two cases are as follow: (1) $\mathcal{FRC}(\_, 5292, 1)$ and (2)

$$\begin{gathered}\mathcal{FRC}s = \{\\ \mathcal{FRC}_1(\_, 1008, 0.07), \mathcal{FRC}_2(\_, 2016, 0.19),\\ \mathcal{FRC}_3(\_, 4032, 0.28), \mathcal{FRC}_4(\_, 8064, 0.46)\}.\end{gathered}$$



(a) Scenario with 4 $\mathcal{FRC}$s, $\mathbb{C} = 0.5$.

(b) $\partial Claims$ and $nextClaim$, $\mathbb{C} = 0.5$.

(c) $\partial Claims$ normalized by ρ, $\mathbb{C} = 0.5$.

(d) Scenario with 4 $\mathcal{FRC}$s, $\mathbb{C} = 0.7$.

(e) $\partial Claims$ and $nextClaim$, $\mathbb{C} = 0.7$.

(f) $\partial Claims$ normalized by ρ, $\mathbb{C} = 0.7$.

(g) Scenario with 4 $\mathcal{FRC}$s, $\mathbb{C} = 0.9$.

(h) $\partial Claims$ and $nextClaim$, $\mathbb{C} = 0.9$.

(i) $\partial Claims$ normalized by ρ, $\mathbb{C} = 0.9$.

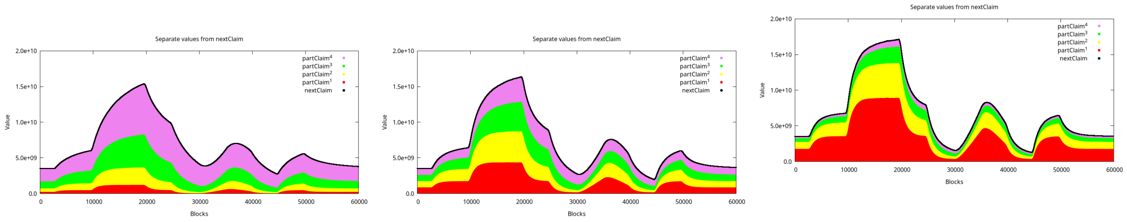
Figure 6.2: Experiment II investigating various $\mathbb{C}$s in the setting with multiple $\mathcal{FRC}$s with their corresponding $\lambda = \{1008, 2016, 4032, 8064\}$ and $\rho = \{0.07, 0.14, 0.28, 0.51\}$. $\partial Claims$ represents contributions of individual $\mathcal{FRC}$s to the total reward of the miner (i.e., its $nextClaim$ component).

We can easily verify that the effective_λ of 4 $\mathcal{FRC}$s is the same as in a single $\mathcal{FRC}$ using Equation 6.1: $0.07 * 1008 + 0.19 * 2016 + 0.28 * 4032 + 0.46 * 8064 = 5292$.

We conducted this experiment using a custom fee evolution scenario involving mainly linearly increasing/decreasing fees in the mempool (see Figure 6.4a), and we set $\mathbb{C}$ to 0.7 for both configurations. The new scenario of the transaction fees evolving in the mempool was chosen to contain extreme changes in fees, emphasizing possible differences between the setups. We are aware of this not being the most accurate real world comparison, but the main focus of this experiment was to display a scenario with the highest difference.

6.4.2 Results

In Figure 6.4b, we show the relative difference in percentages of *nextClaim* rewards between the settings of 4 $\mathcal{FRC}$s versus 1 $\mathcal{FRC}$. It is clear that the setting of 4 $\mathcal{FRC}$s in contrast to a single $\mathcal{FRC}$ provided better reward compensation in times of very low fees value in the mempool, while it provided smaller reward in the times of higher values of fees in the mempool. Therefore, we concluded that it is not possible to replace a setup of multiple $\mathcal{FRC}$s with a single one, even though in a more realistic environment we might see much smaller differences due to more balanced spread of fees across time in mempool. This can be partially seen as comparison between Figure 6.1 and Figure 6.2 where we intentionally run with very similar effective_λ.



(a) ρ correlating with λ.

(b) ρ equal for every $\mathcal{FRC}$.

(c) ρ negatively correlating with λ.

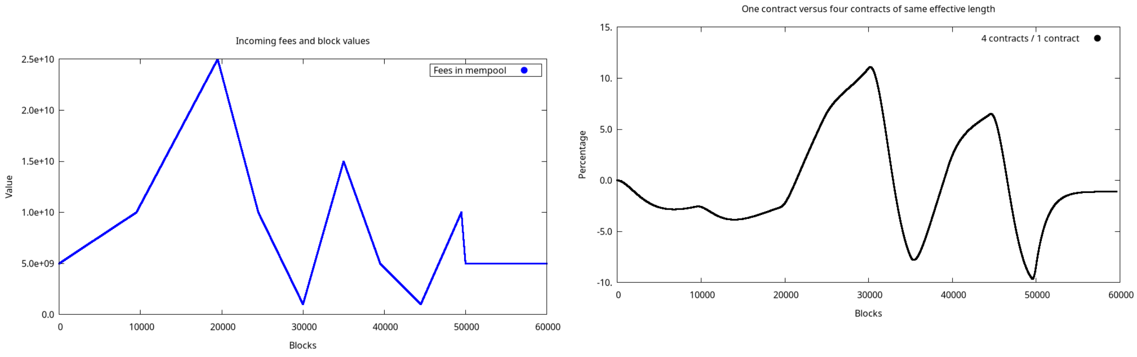
Figure 6.3: Experiment II – multiple $\mathcal{FRC}$s using various distributions of ρ and their impact on $\partial Claim$, where $\mathbb{C} = 0.7$.

6.5 Experiment IV – Historical data of miner's reward fluctuation

This experiment tries to directly target fluctuation of miner's revenue presented in Section 4.1. We will observe similarly to previous experiments, the baseline (i.e. miner's reward without our solution) compared to the rewards received by the miner with our solution. We will be recreating the real world data from Figure 4.1 into our custom scenario, and we will examine multiple single contract $\mathcal{FRC}$s with different lengths.

6.5.1 Methodology

The baseline scenario is represented once again with the "*Fees in mempool*" series. We have recreated the Figure 4.1 directly, meaning each day consists of 144 blocks (as per Bitcoin) and each day will have fixed fee income as per the charts from 2017. Each day was



(a) A custom fee scenario for Experiment III.

(b) A relative difference in $nextClaim$ between 4 $\mathcal{FRC}$s and a single $\mathcal{FRC}$.

Figure 6.4: Experiment III comparing 4 $\mathcal{FRC}$s and 1 $\mathcal{FRC}$, both configurations having the same effective_λ.

calculated by the following formula, while we were still aiming for the 50BTC average fees as previously.

$$\overline{fees} = 50BTC$$
$$B_{dayInWeek}.fees = \frac{\frac{percent_{dayInWeek}}{100}}{\frac{1}{7}} * \overline{fees}$$

And as mentioned, the whole 144 blocks would have the same value for the corresponding day of $B_{dayInWeek}.fees$. In total, we run 3 simulations, each consisting of one simulation run for one setup of $\mathcal{FRC}$ and simulated total of 49 weeks. The data represented in Figure 6.5 were taken as the results from simulated week 46 & 47. The total number of weeks was selected large enough to show clear data, without any potential adjustments of $\mathcal{FRC}$s during the first weeks, of how an adapted $\mathcal{FRC}$ would react to changes.

We run the experiment with 3 following $\mathcal{FRC}$s:

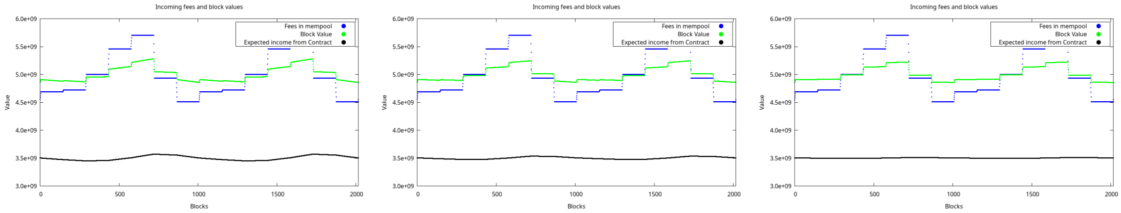
$$\mathcal{FRC}^1 = \{\mathcal{FRC}_1(\_, 1008, 1)\}$$
$$\mathcal{FRC}^2 = \{\mathcal{FRC}_1(\_, 2016, 1)\}$$
$$\mathcal{FRC}^3 = \{\mathcal{FRC}_1(\_, 8064, 1)\}$$

where the initial value of $\mathcal{FRC}$s (i.e. $\mathcal{FRC}.\nu$) is calculated as defined in Equation 5.12.

6.5.2 Results

As we present in Figure 6.5 it is clear our solution influences the historical data as we would expect. Where the fluctuations between days are much less visible and the jumps are mostly impacted solely by the $\mathbb{M}$ ratio. This is visible from the "*Expected income from Contract*" series, which is fluent and averages the incoming fees. We once again demonstrate how longer λ does more impactful averaging.

This experiment was to show the direct effect when trying to mitigate only one fluctuation's time period. In reality, we would expect to have setup of multiple $\mathcal{FRC}$s with different ρ so the real averaging might be slightly smaller.



(a) $\mathcal{FRC}^1$ with $\lambda = 1008$. (b) $\mathcal{FRC}^2$ with $\lambda = 2016$. (c) $\mathcal{FRC}^3$ with $\lambda = 8064$.

Figure 6.5: Experiment IV – Direct recreation of Figure 4.1 into a scenario against our solution with $\mathcal{FRC}$s.

6.6 Experiment V – Undercutting attack with $\mathcal{FRC}$s used

We focused on reproducing the experiment from Section 5.5 of [5]. We were searching for the minimal ratio of Default-Compliant miners, at which the undercutting attack is no longer a profitable strategy. Default-Compliant miners are honest miners in a way that they follow the rules of the consensus protocol, such as building on top of the longest chain. In case we have two chains, Default-Compliant miner mines on the older chain from his point of view.

We executed several simulations, each consisting of multiple games (i.e., 300k as in [5]) with various fractions of Default-Compliant miners. From the remaining miners, we evenly created *learning miners*, who learn on the previous runs of games and switch with a certain probability to the best mining strategy out of the following.

- PettyCompliant: This miner behaves as Default-Compliant except one difference. In the case of seeing two chains, he does not mine on the oldest block, but rather the most profitable block for him. Thus, this miner is not the (directly) attacking miner and with no malicious miners forking the chain, he behaves most of the time exactly the same as Default-Compliant. However, existence of this type of miner supports the undercutting miners and disincentivizes the occurrence of Default-Complaint miners.
- LazyFork: This miner checks which out of two options is more profitable: (1) mining on the longest-chain block or (2) undercutting that block. In either way, he leaves half of the mempool fees for the next miners, which prevents another LazyFork miner to undercut him.
- Function-Fork() The behavior of the miner is parameterized with a function f(.) expressing the level of his undercutting. The higher the output number, the less reward he receives and more he leaves to incentivize other miners to mine on top of his block. This miner undercuts every time he forks the chain, and he has to undercut with his function f(.) parameter.

6.6.1 Methodology

With the missing feature for difficulty re-adjustment (present in our work as well as in [5]) the higher orphan rate occurs, which might directly impact our $\mathcal{FRC}$-based approach. If the orphanage rate is around 40%, roughly corresponding to reproducing the original experiment [5] with unchanged simulator, our blocks would take on average 40% longer to be created, increasing the block creation time (i.e., time to mine a block). In reality the blockchain would adjust for this with lowering the difficulty to increase the production of blocks, however this is not part of the simulator and was not built with this in mind. With that, the mean blocktime differs from what we expect when starting the simulation with our parameters. This does not affect the original simulator as much, as there are no $\mathcal{FRC}s$ that would change the total reward for the miner who found the block.

To extrapolate the total value in the final blockchain in the original simulator, i.e., sum of all rewards of miners of all the transaction fees, in the longest chain is:

$$(head().timestamp - genesis().timestamp) * feeRate,$$

where the $feeRate$ in this context means something different than previous mentions. Here the $feeRate$ corresponds to constant fees incoming every second into the mempool (as in the original simulator), $head()$ returns the highest block from the longest chain, and $genesis()$ returns the genesis block.

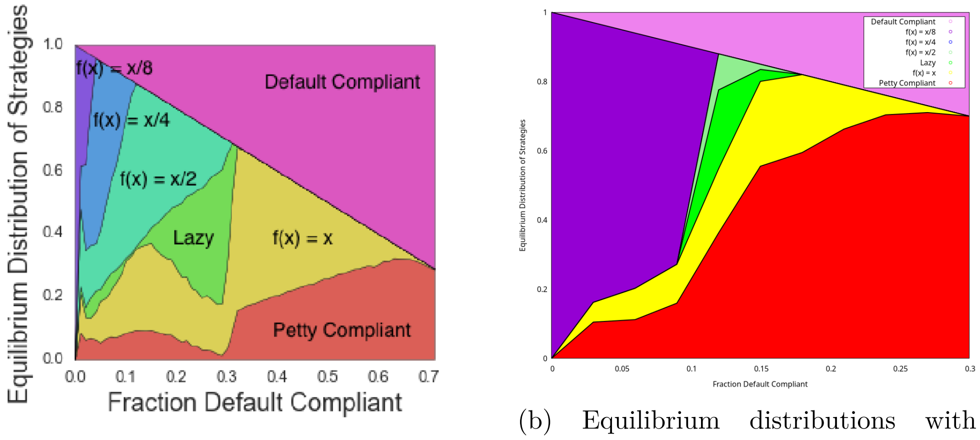
Nevertheless, this is not true for $\mathcal{FRC}$-based simulations, as the initial setup of $\mathcal{FRC}s$ is calculated with $\overline{fees} = 50$ BTC (as per the original simulations). However, with longer block creation time and transaction fees being calculated from it, the value of $\overline{fees}$ also changes. Without any adjustments, this results in $\mathcal{FRC}s$ initially paying smaller reward back to the miner before they are saturated. To mitigate this problem, we increased the initial values of individual $\mathcal{FRC}$s by the orphanage rate from the previous game before each run.

This results in very similar conditions, which can be empirically verified by comparing the final value in the longest chain of our simulation versus the original simulations. We decided to use this approach to be as close as possible to the original experiment. This is particularly important when the fullMempool parameter Paragraph 6.1.4 is equal to `False` (see Section 6.1), which means that the incoming transaction fees to mempool are calculated based on the block creation time. In our simulations, we used the following parameters: 100 miners, 10 000 blocks per game, 300 000 games (in each simulation run), exp3 learning model, and $\mathbb{C} = 0.7$. Modeling of fees utilized the same parameters as in the original paper [5]: the fullMempool parameter disabled, a constant inflow of 5 000 000 000 Satoshi (i.e., 50 BTC) every 600s. We have tried to recreate the original experiment as closely as possible, but we have picked the value $\mathbb{C}$ rather empirically based on previous experiments. We consider this value rather relaxed.

For more details about the learning strategies and other parameters, we refer the reader to [5].

Setup of $\mathcal{FRC}s$

Since we have a steady inflow of fees to the mempool, we do not need to average the income for the miner. Therefore, we used only a single $\mathcal{FRC}$ defined as $\mathcal{FRC}$(7 056 000 000 000, 2016, 1), where the initial value of $\mathcal{FRC}.\nu$ was adjusted according to Equation 5.12, assuming $\overline{fees} = 50$ BTC. Even though the $\mathcal{FRC}.\lambda$ influences the final reward for the miner, we do not think it makes drastic changes to the results. However, it is chosen rather small to leave more room for slight fluctuations. In next runs of any game, $\mathcal{FRC}.\nu$ was increased by the orphanage rate from the previous run, as mentioned above, to ensure a rather consistent environment.



(a) Equilibrium distributions without $\mathcal{FRC}$.

(b) Equilibrium distributions with $\mathcal{FRC}$.

Figure 6.6: Stacked area chart showing equilibrium distributions of different mining strategies. The number of Default-Compliant miners in our $\mathcal{FRC}$ approach is ∼30% (in contrast to ∼ 66% of [5]).

6.6.2 Results

The results of this experiment are depicted in Figure 6.6, and they demonstrate that with our approach using $\mathcal{FRC}$s, we lowered down the number of Default-Compliant miners (i.e., purple meeting red) from the original 66% to 30%. This means that the profitability of undercutting miners is avoided with at least 30% of Default-Compliant miners, indicating a more robust result. This implies a more robust system against undercutting attacks, which were presented in the paper [5], on the protocol level.

Chapter 7

Security Analysis and Discussion

7.1 Contract-drying attack

This is a new attack that might potentially occur in our scheme; however, it is the abusive attack aimed at attacking the functionality of our scheme and not on maximizing the profits of the adversary. In this attack, the adversary aims at getting his reward only from $\mathcal{FRC}$s and does not include transactions in the block (or includes only a number of them). This might result in slow drying of the funds from $\mathcal{FRC}$s and would mean less reward for future honest miners. Moreover, the attacker can mine in well times of higher saturation of $\mathcal{FRC}$s and after some time decide to switch off the mining. This might cause a deterioration in profitability for honest miners, consequently leading to deteriorated security w.r.t., undercutting attacks. With that, it might possibly occur in cycles with times of higher saturation (i.e. start of the attack) and lower saturation of $\mathcal{FRC}$s (i.e. the end of the attack).

The attacker successfully executing this attack for multiple blocks would likely lead to increased number of transactions in the mempool since the attacker includes less transactions in the block (thus more in the mempool) than the honest miners. The users of the blockchain might opt for higher transactions fees to ensure they will get eventually included in the next "healthy" block. Therefore, if an honest miner mines a block, he gets higher reward and at the same time deposits a higher amount from transaction fees to $\mathcal{FRC}$s, which indicates a certain self-regulation of our approach, mitigating this kind of attack.

Additionally, we can think of lowering the impact of this attack by rewarding the miner with the full $nextClaim_{[H+1]}$ by $\mathcal{FRC}$s only if the block contains enough transaction fees (e.g., specified by the network-regulated minimum threshold). It is based on collected fees rather than amount of transactions, as the miner might create artificial transactions with small fees. However, this assumes that there is always a reasonable amount of fees in the mempool, which might not be the case all the time and might result in a situation where the miners temporarily stop mining if there is not enough transactions to mine. However, we do not consider this to be a realistic threat to our solution. Nonetheless, it would require more research to investigate this solution or a better solution mitigating this type of potential abusive attack, which we left for future work.

7.2 Possible improvements

$\mathcal{FRC}$ can contain the parameter enabling the interval of the possible change in the reward paid by $\mathcal{FRC}^x$ (i.e., $nextClaim^x_{[H+1]}$) from the median of its value (computed over λ or perhaps a new parameter for the length). If $nextClaim^x_{[H+1]}$ would drastically increase from its median value as significantly more fees would come into the mempool, then $\mathcal{FRC}^x_{[H]}$ would reward the miner with a certain value (specified by the parameter) from the interval $\langle average, nextClaim^x_{[H+1]}\rangle$ instead of the full $nextClaim^x_{[H+1]}$. This would be particularly useful for $\mathcal{FRC}$s with a small λ parameter. The parameter used for sampling might contain a stochastic function (e.g., exponential) attributing a higher likelihood of getting the values not far from the median. This is somewhat similar to how difficulty adjustments in Bitcoin work, where in case of difficulty the maximum it can change between epochs is 300%. However, we left the evaluation of this technique to future work.

7.3 Epoch like $\mathcal{FRC}$s

$\mathcal{FRC}$ can instead of sliding window work similarly to how mining epochs in PoW blockchains work. Providing the same reward for the whole time of the epoch of the corresponding $\mathcal{FRC}$. Where in the beginning of each epoch for individual $\mathcal{FRC}$s we would calculate the:

$$\partial Claim^{\mathcal{FRC}_{[E]}}_{[E+1]} = \frac{\mathcal{FRC}_{[E]}.\nu}{\mathcal{FRC}_{[E]}.\lambda}$$

(please note the change to using E – epoch of the particular $\mathcal{FRC}$ corresponding to it's λ) would instead of being calculated for individual blocks as in Chapter 5, calculate it for the whole λ of the corresponding $\mathcal{FRC}$.

This would mean the $\partial Claim^{\mathcal{FRC}_{[E]}}_{[E+1]}$ would be calculated once and for the whole duration of λ blocks would stay the same. While at the same time all the reward coming to the $\mathcal{FRC}$ would be collected for the next epoch.

This solution would not reflect the changes better than current solution especially for $\mathcal{FRC}$s with longer λs, however it would lead to even more predictable rewards. However, this would come at a cost of more complex logic and memory footprint for $\mathcal{FRC}$s. It would need to have additional parameter for holding a state calculating how deep inside an epoch the individual $\mathcal{FRC}$ and when to recalculate the $\partial Claim^{\mathcal{FRC}_{[E]}}_{[E+1]}$. In addition, another parameter holding the $\partial Claim^{\mathcal{FRC}_{[E]}}_{[E+1]}$ for this duration would be needed.

Nonetheless, we believe the next idea presented in Section 7.4 would make more sense in regard to this solution.

7.4 Adjustment of mining difficulty

If the PoW blockchain with the longest chain fork-choice rule uses transaction-fee-based regime, the profitability of miners might be more volatile. This assumption comes from the fact the blockchain incentivizes miners only by transaction fees and with the problem of fluctuation described in Section 4.1, which can lead to varying time between blocks, decreased security w.r.t. undercutting attacks, etc. Although our solution with $\mathcal{FRC}$s helps in mitigation of this problem, we propose another functionality that resides in adjusting the mining difficulty based on the total collected fees during the epoch. In detail, the difficulty can be increased with higher fees collected from transactions during the epoch and vice versa. Expecting a higher influx or departure of miners as the rewards from $\mathcal{FRC}$s would change. This means, that if we calculate profitability solely in native currency and the transaction fees increases during the epoch, we can partially adjust the difficulty next to the current difficulty adjusting mechanism. This would especially make sense in combination with Section 7.3. Further research would be needed to evaluate this proposition.

7.5 Allowing soft fork implementation

With many changes to the protocol, hard forks are required, which makes many communities disregard such changes. We believe so, as this splits the original protocol into two protocols, and it is up to the users to choose the one they would like to stay on. This split can be seen in examples such as Bitcoin or Ethereum having many hard forked versions. Leading it to years long debates about the block size wars on Bitcoin protocol. We believe our solution should not encounter this problem, which results in higher chance of being adopted.

It should be possible to implement on most of current blockchains, where we would create a smart contract [20] including the logic of our $\mathcal{FRC}$-based solution. Even the multi contract $\mathcal{FRC}$s could be possibly created within a single smart contract, which would lead to smaller memory and transaction footprint on the final blockchain. Furthermore, we would create protocol rules, which would enforce miners to send $\mathbb{C}$ fraction from all collected fees into $\mathcal{FRC}$(s). This would result in old nodes still being able to process new transactions including a smart contract execution as valid, while miners with old consensus rules would get their blocks rejected by the updated nodes – a soft fork.

7.6 So called out-of-band fees

Out-of-Band fees are referred to any fees being paid outside the protocol, e.g. transaction fee not being paid in BTC in case of Bitcoin.

This can be utilized by a user, such as big exchanges, trusted wallet providers etc. with an agreement with a miner, such as big mining pool. Where the user would contact miner of his transaction, which will have none to very low transaction fee. Under normal circumstances such a transaction would not make it into the block as it's feerate Paragraph 3.7 is too low for a rational miner to include it. However, if the user has a mutual agreement with some miners, that upon including such transaction they would pay them out-of-band fees one of the miners will include this transaction. Then the user will pay them this out-of-band fee, which can be away from the protocol, i.e. traditional fiat currency payment, or he can create a payment on Bitcoin network directly to the miner's public key.

Even though this is not used widely nowadays, because of no general advantage, the implementation of redistribution contracts would lead to higher utilization of such schemes. Furthermore, it would directly lead to higher centralization of miners into the biggest pools, as a user trying to use out-of-band fees would contact only the biggest mining pools and smaller pools would be loosing on this revenue. Also, a user not utilizing out-of-band fees might be overpaying in transaction fees in order to get their transaction included compared to user directly paying out-of-band fees. In the process, destroying the healthy free fee market that is being currently used.

Preferring out-of-band fees. As with redistribution contracts the miner would be losing his income in favor of a future miner, he might opt to receive a bit smaller fee if this fee goes directly to him, i.e. circumventing $\mathbb{C}$ fraction. Also, this would benefit the user as he would be paying smaller fee overall as the miner would prefer out-of-band fees.

7.7 Utilization of Fee-Redistribution Contracts

However, it is still possible to use redistribution contracts on schemes, where out-of-band fees would not be possible. One such example can be found on Ergo blockchain [8], where a new novel feature of Storage Rent fee was introduced.

This means each UTXO box [16], which was not spent in over four years, is a subject to Storage Rent fee, where the miner can claim some value in Ergs (or destroy the box and keep its tokens) defined by protocol. Presented as a potential solution to bloating, resulting in additional revenue for miners. However, this fee is not subject of out-of-band fees, as the user paying this fee has forgotten to move his box in order to prevent the Storage Rent fee and for the miner it would be extremely hard to contact owner of box, soon to be subject of Storage Rent fee, just by his address.

We also provide an example code snippet for general redistribution contracts on Ergo blockchain for observation in Appendix A.

Chapter 8

Conclusion

The main motivation for this paper on how to solve possible bitcoin's future problem and a motivation for blockchains of future to consider this Proof of Concept into their design. In this work, we focused on three problems related to transaction-fee-based regime blockchains with the longest chain fork choice rule: (1) the instability of mining rewards, (2) the possibility of undercutting attacks, and (3) the mining gap. However, this solution is not exclusive only to PoW blockchains or collecting transaction fees as shown in Appendix A but further research would be needed for blockchains such as Proof-of-Stake based etc.

To mitigate these problems, we proposed the approach approximating a moving average based on the fee-redistributions contracts that accumulate a certain fraction of transaction fees and at the same time reward the miners from their reserves. In this way, the miners are sufficiently rewarded even at the time of very low transaction fees, such as the beginning of the mining round, entering the mining protocol by new miners, market deviations, etc.

In order to create this work we have studied not only the work of Carlsten et al. [5], but also got ourselves familiar with different solutions. We have decided to come up with a solution enforced by the protocol rather than relying on an avoidance-like solution. Based on our setup, we came up with a solution, which we proposed under Chapter 5 and vigorously executed multiple experiments in Chapter 6. We observed the behavior of our solution with different parameters and their combinations. We have tested most scenarios with fixed income per block (i.e. fullMempool Paragraph 6.1.4 set to `True`) as the block limit size is the hard constraint in most blockchains not allowing to exceed a certain amount of transactions. This can be usually verified on website such as [17].

Besides directly replicating the undercutting attack, our approach brings a higher tolerance to this particular type of attack, and increases the minimal threshold of DEFAULT-COMPLIANT miner that strictly do not perform undercutting attack from 66% reported in state-of-the-art to 30%.

We believe it is possible to use this solution on most blockchains with a soft fork, which leads to higher chance of being adopted by the underlying protocols. Therefore, making mining on transaction-fee-based regime blockchains more sustainable and predictable. Resulting in more healthy environment for all honest participants of the particular blockchain.

However, with existence of so called out-of-band fees this solution might lead to higher centralization of miners in mining pools, which should be avoided. Thus, unless a solution mitigating usage of out-of-band fees, to circumvent sending funds to fee-redistribution contracts, is not discovered, fee-redistribution contracts should not be recommended as a solution for transaction-fee-based regimes.

Bibliography

[1] bits.monospace.live. *Monospace live.* 2023. Available at: `https://bits.monospace.live/`.

[2] Bjercke, B. and Finlow Bates, K. Decoupling Bitcoins from Their Transaction History Using the Coinbase Transaction. march 2020.

[3] Bjercke, B. and Finlow Bates, K. Decoupling Bitcoins from Their Transaction History Using the Coinbase Transaction. 2020. Available at: `https://luxsci.com/blog/understanding-blockchains-and-bitcoin-technology.html`.

[4] Budinský, R. *Changed Bitcoin mining simulator* [online]. 2023. Available at: `https://github.com/The-Huginn/mining_simulator`.

[5] Carlsten, M., Kalodner, H., Weinberg, S. M. and Narayanan, A. On the instability of bitcoin without the block reward. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security.* 2016, p. 154–167.

[6] cdecker. *With 100% segwit transactions, what would be the max number of transaction confirmation possible on a block?* [online]. [cit. 2023-11-02]. Available at: `https://bitcoin.stackexchange.com/a/59501`.

[7] Daian, P., Goldfeder, S., Kell, T., Li, Y., Zhao, X. et al. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In: IEEE. *2020 IEEE Symposium on Security and Privacy (SP).* 2020, p. 910–927.

[8] Ergoplatform.org. *Ergo blockchain.* 2022. Available at: `https://ergoplatform.org/en/`.

[9] Eyal, I., Gencer, A. E., Sirer, E. G. and Van Renesse, R. {Bitcoin-NG}: A scalable blockchain protocol. In: *13th USENIX symposium on networked systems design and implementation (NSDI 16).* 2016, p. 45–59.

[10] Gong, T., Minaei, M., Sun, W. and Kate, A. Towards overcoming the undercutting problem. In: Springer. *International Conference on Financial Cryptography and Data Security.* 2022, p. 444–463.

[11] Homoliak, I., Venugopalan, S., Reijsbergen, D., Hum, Q., Schumi, R. et al. The Security Reference Architecture for Blockchains: Toward a Standardized Model for Studying Vulnerabilities, Threats, and Defenses. *IEEE Communications Surveys & Tutorials.* 2021, vol. 23, no. 1, p. 341–390. DOI: 10.1109/COMST.2020.3033665.

[12] Houy, N. The economics of Bitcoin transaction fees. *GATE WP*. 2014, vol. 1407.

[13] Hum, Q., Tan, W. J., Tey, S. Y., Lenus, L., Homoliak, I. et al. CoinWatch: A clone-based approach for detecting vulnerabilities in cryptocurrencies. In: IEEE. *2020 IEEE International Conference on Blockchain (Blockchain)*. 2020, p. 17–25.

[14] Kalodner, H. *Bitcoin mining simulator* [online]. 2015. Available at: `https://github.com/citp/mining_simulator`.

[15] Khatwani, S. *Bitcoin nonce*. 2023. Available at: `https://themoneymongers.com/bitcoin-nonce/`.

[16] Lopp, J. *Unspent transactions outputs in Bitcoin*. 2015. Available at: `https://blog.lopp.net/the-challenges-of-optimizing-unspent-output-selection/`.

[17] mempool.space. *Mempool Statistics*. 2022. Available at: `https://mempool.space/`.

[18] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*. 2008, p. 21260.

[19] Rodriguez, G. *What is blockchain*. 2023. Available at: `https://money.com/what-is-blockchain/`.

[20] Szabo, N. The idea of smart contracts. *Nick Szabo's papers and concise tutorials*. 1997, vol. 6, no. 1, p. 199.

[21] Tsabary, I. and Eyal, I. The gap game. In: *Proceedings of the 2018 ACM SIGSAC conference on Computer and Communications Security*. 2018, p. 713–728.

[22] Tucker, A. W. and Luce, R. D. *Contributions to the Theory of Games*. Princeton University Press, 1959.

[23] Wiki, B. *Miner Fees*. 2022. Available at: `https://en.bitcoin.it/wiki/Miner_fees`.

[24] Zhou, D., Ruan, N. and Jia, W. A robust throughput scheme for bitcoin network without block reward. In: IEEE. *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. 2019, p. 706–713.

Appendix A

Fee-Redistribution Contracts on Ergo Blockchain

A.1 Brief introduction into Ergo

Ergo is eUTXO [16] based blockchain, i.e. similar to Bitcoin, however it is quasi Turing complete. It allows writing smart contracts and these contracts are similarly to Bitcoin guarding scripts of the particular UTXO box. Each box has few registers, which can be used to store information. One of the registers for example contains the guarding script, another contains information about tokens in a box, e.g. NFTs. Furthermore, each script can be created with parameters inside it.

When one wants to spend a box he has to off-chain execute the smart contract, i.e. guarding script, which can enforce what outputs must be created. Writing smart contracts is a bit different from platforms such as Ethereum. You write the assumptions, outcomes and conditions, that have to be met. This resembles functional programming opposed to procedural or OO programming as it is based on sigma calculus and reference client is written in Scala language. Resulting in smart contracts being mostly propositions evaluating to true or false in the end.

Note: on one address multiple same boxes, i.e. with the same guarding script, might be. To identify the correct one, we use NFTs to uniquely identify the wanted one. That is why in the code example NFT token is used. For more details, such as Blockchain context variables, follow this link.

A.2 Smart contract overview

The Contract ensures whenever it is spent a new box with the same guarding script is created as first output from transaction, preserving the identifying NFT of the box Paragraph A.1. Furthermore, it guarantees it can be spent only once per block and is spent in the currently mined block, i.e. not in the past or future. Then it checks how much a miner has claimed and how much value is in the next box with redistribution contracts.

We also ensure the individual redistribution contracts have their values updated correctly.

A.3 Implementation

```
// fetch first transaction output box
val contractOut = OUTPUTS(0)

// ensure nft identifier is preserved
val correctNftOut = contractOut.tokens(0)._1 == redistributionNft

// created for current blockchain height
val heightCorrect = contractOut.creationInfo._1 == HEIGHT

// box is spendable only once per block
val heightIncreased = HEIGHT > SELF.creationInfo._1

// script is preserved in the output box
val correctProposition =
    contractOut.propositionBytes == SELF.propositionBytes

// extracting Erg value going to miner
val minerReward =
    OUTPUTS.slice(1, OUTPUTS.size - 1)
        .fold(0L, { (sum: Long, output: Box) =>  output.value + sum })

// register holding values of individual redistribution contracts
val values = SELF.R4[Coll[Long]].get


val indices = values.indices

// register holding update values of individual
//  redistribution contracts after miner's claim
val newValues = contractOut.R4[Coll[Long]].get

// we calculate nextClaim from current state of redistribution contracts
val claimableReward = indices
    .fold(0L, { (sum: Long, i: Int) =>  sum + values(i) / params(i)._1 })

// miner has claimed at most nextClaim, i.e. claimableReward
val correctMinerClaim = minerReward <= claimableReward

// this should always be positive
val collectedFees = contractOut.value - (SELF.value - minerReward)

// ensuring values of individual
//  redistribution contracts are correctly updated
val correctValues = allOf(Coll({
    indices.forall({ (i: Int) => ({
        // how much miner took from individual contracts,
        //  ratio has to be kept
        val partialClaim = (minerReward * params(i)._2) / 100

        // we calculate what will be added to redistribution contract
        val addedValue = (collectedFees * params(i)._2) / 100
        // ensure updated values in redistribution contracts are correct
        newValues(i) >= values(i) - partialClaim + addedValue
        })
    })
}))

// all previous propositions have to hold true at the same time
sigmaProp(
    allOf(
        Coll(
            correctNftOut,
            heightCorrect,
            heightIncreased,
            correctProposition,
            correctMinerClaim,
            correctValues
        )
    )
)
```

Listing A.1: Ergoscript example of redistribution contracts

Appendix B

Contents of Included Storage Device

Attached CD drive has following structure:

- **src/** - source code of simulator
  - **Readme.md** - User's manual for installation and usage
  - **LICENSE.txt** - licenses
  - **Makefile** - for compiling sources
  - **feeContracts.json** - Configuration file for $\mathcal{FRC}$-s
  - **feeSimulation.json** - Configuration file for transaction fees
  - **BlockSim/** - core source codes for simulator
  - **StratSim/** - source codes for running simulations
  - **scripts/** - different scripts for conducting experiments
  - **repo.txt** - link to Github repository of this simulator
  - **strat** - executable for simulation produced by Makefile
- **docs/** - source codes for generating this pdf, including Makefile
- **xbudin05.pdf** - this pdf file