

Pay To Win

Algorithmic Incentive Manipulation Attacks on Permissionless Cryptocurrencies

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

eingereicht von

Dipl.-Ing. Aljosha Judmayer, BSc

Matrikelnummer 0826392

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.-Prof. Dipl.-Ing. Mag.rer.soc.oec. Dr. techn. Edgar R. Weippl

Diese Dissertation haben begutachtet:

Rainer Böhme

Bryan A. Ford

Wien, 18 Juli, 2023

Aljosha Judmayer

Pay To Win

Algorithmic Incentive Manipulation Attacks on Permissionless Cryptocurrencies

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Dipl.-Ing. Aljosha Judmayer, BSc

Registration Number 0826392

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.-Prof. Dipl.-Ing. Mag.rer.soc.oec. Dr. techn. Edgar R. Weippl

The dissertation has been reviewed by:

Rainer Böhme

Bryan A. Ford

Vienna, 18th July, 2023

Aljosha Judmayer

Erklärung zur Verfassung der Arbeit

Dipl.-Ing. Aljosha Judmayer, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 18 Juli, 2023

Aljosha Judmayer

Acknowledgements

When finishing a dissertation, one can be grateful for many things: your family and friends, the environment that made it all possible, professors, lecturers and teachers you encountered on the way, the ability to engage into inspiring discussions with scientists that value and practice the exchange of ideas, the possibility to stand on the shoulders of giants in your respective field, numerous grants, the tax payer who funded most of them and so forth.

These are all aspects which deserve acknowledgement, but here I want to dedicate the remaining space to the people who deserve it (subjectively) the most, also due to the temporal and social proximity. I want to thank my dear colleagues (in various institutions SBA Research, TU Vienna and University of Vienna) some of which have become close friends over the years. As Phd students and post docs they belong to the pillars of academia. Their motivation and aspiration is the fuel which drives the progress in acquiring and disseminating knowledge alike.

I was lucky to work with a lot of remarkable and inspiring people over the years, acknowledging each and everyone would require a document on its own. Therefore, I will only explicitly name those I spent the most of my productive and creative time over the recent years (in alphabetical order ;)

I want to thank *Georg Merzdownik*, who has profound systems- and IT security knowledge and an incomparable talent to break things, which makes him the perfect advocatus diaboli when discussing or scrutinizing literally anything. He is also always available and helps when there is a bug, or an IT problem to track down. His desire for lean interfaces and performance also makes him the perfect guide who always recommends the right tool for the job (at least according to my taste). Especially in the context of GNU/Linux and FOSS I owe him some good (not to say life-changing) recommendations and configurations.

I want to thank *Philipp Schindler*, who has an amazing ability to grasp complex things and an inspiring level of self-confidence when searching for a solution to a difficult problem. I owe his helpfulness and pragmatism a lot of valuable advice regarding developing and prototyping in Python. Combined with his capability to focus and concentrate on a chosen task, working with him is like having a productivity booster rocket. His urge and motivation to make great things and excel also stretches to other areas of interest, which brought me into contact with 3D printing.

I want to thank *Nicholas Stifter*, who has a profound knowledge on- and deep understanding of distributed systems and cryptocurrencies which he is willing to share with everyone at any time. This vast background knowledge and his readiness to help is an invaluable assistance when searching for related literature, a citation or a reference in this context. His passion to discuss and examine things from various different angles often paved the way for incredible ideas and insights of varying magnitude, which sometimes happen as a byproduct. Even the ideas, which we were not able to execute in our small group, or simply forgot about - only to recognize them again later in another paper - would easily fill an entire dissertation on its own. While being a critical thinker always capable to self-reflect, he also gets in touch with the community and has kept a hands-on-mentality regarding cryptocurrencies. I guess his openness and the ability to step back and take a broader perspective also is a good substrate for his various different hobbies, which made me learn about trees, brewing beer, diving, pizza and countless other things.

Thank you guys, it really was a pleasure.

Kurzfassung

Nakamoto Consensus (NC) stellt die Grundlage für die meisten *Permissionless* Cryptocurrencies dar. Die genauen Regeln können von System zu System unterschiedlich sein, aber alle Varianten erfordern einen Beweis dafür, dass eine gewisse Ressourcenaufwand erbracht wurde um einen neuen Systemzustand vorzuschlagen. Diese Resource kann Rechenleistung und Energie sein, wie beispielsweise in Proof-of-Work (PoW), oder ein gewisser Betrag an Cryptocurrency-Einheiten wie in Proof-of-Stake (PoS). Um zur Teilnahme anzuregen, belohnen Permissionless-Cryptocurrency die Teilnehmer mit systemeigenen Werteinheiten.

Eine Kernfrage dabei ist es, wie das korrekte Verhalten der Teilnehmern incentiviert werden kann. Obwohl Cryptocurrencies in den letzten Jahren intensiver erforscht wurden, gibt es immer noch blinde Flecken und unbeantwortete Fragen in Bezug auf das Verständnis der darunterliegenden Sicherheitsgarantien und Konstruktionen. Eine dieser Fragen betrifft die Sicherheit unter Berücksichtigung von Attacken welche automatisiert abrufbare Belohnungen algorithmisch zusichern für den Nachweis von Aktionen welche vom Angreifer gewünscht wurden.

In dieser Arbeit werden Angriffe, welche auf die Anreize der Teilnehmer abzielen, unter einer neuen Kategorie von Angriffen namens *Algorithmic Incentive Manipulation* (AIM) zusammengefasst. AIM stellt programmatisch Belohnungen oder Drohungen in Aussicht, welche die Anreize von ökonomisch rationalen Teilnehmern verändern. Auf diese Weise kann ein Angreifer die Wahrscheinlichkeit erhöhen, dass das angegriffene System einen gewünschten Zustand erreicht. Je nach Motiv des Angreifers ist es nicht notwendigerweise erforderlich, dass er direkt von seinem gewünschten Zustand profitiert. Wenn der gewünschte Systemzustand jedoch zu Gewinnen für den Angreifer führt, können Anteile des Gewinns verwendet werden, um algorithmisch erzwingbare Zusatzzahlungen für die konspirierenden Teilnehmer des AIM-Angriffs durchzuführen. In anderen Worten, der Angreifer zahlt um zu gewinnen, daher "Pay To Win".

Es ist in der Spieltheorie bekannt, dass konspirierende Teilnehmer und Kompensationszahlungen große Herausforderungen für das Mechanism-Design von Spielen darstellen. Im Bezug auf Permissionless Cryptocurrencies sind diese Probleme jedoch noch nicht eindeutig verstanden. Diese Arbeit beschreibt die notwendigen Grundlagen für ein besseres Verständnis. Es wird das Problem beschrieben, verwandte Attacken systematisiert, neue Attacken vorgestellt (inklusive Proof-of-Concept), die Erfolgswahrscheinlichkeit sowie die

Wirtschaftlichkeit von Angriffen evaluiert, sowie ein Beweis erbracht, dass es unmöglich ist in Permissionless-Cryptocurrencies solche Angriffe durch rein technische Maßnahmen zu verhindern. In einer Welt, in der mehrere Cryptocurrencies nebeneinander existieren welche kryptographisch miteinander verflochten werden können, kann die Verfügbarkeit von (externen) Cryptocurrency Ressourcen, welche für aktuelle und potenzielle Teilnehmer relevant sind, nicht ausgeschlossen werden. Daher sollten die erreichbaren Sicherheitsgarantien von Permissionless-Cryptocurrencies, sowie Design-Entscheidungen welche das Problem verstärken, überdacht werden.

Abstract

Nakamoto consensus (NC) lies at the foundation of the prevalent *permissionless* cryptocurrency design. The exact rules differ from system to system, but each variant requires proof that a certain amount of resources was invested in proposing a new system state. These resources, for example, could be computation and energy, as in Proof-of-Work (PoW), or a certain amount of cryptocurrency units, as in Proof-of-Stake (PoS). To incentivize participation, prevalent permissionless cryptocurrencies issue rewards in the form of native currency units.

A key question in this regard is how to incentivize the *honest* behavior of participants. Although cryptocurrencies have been a prominent research object in recent years, the security guarantees of the underlying constructions still contain blind spots and unanswered questions. One of these questions is the security of such systems against attacks which promise automatically claimable rewards, that are algorithmically assured, for provable attacker-desired actions of participants.

In this thesis, we summarize related attacks targeting the incentives of participants under a new attack category termed *algorithmic incentive manipulation* (AIM). AIM programmatically offers rewards, or issues threats, which change the incentives of economically rational players. Thereby, an attacker can increase the chance that the targeted system reaches a favored state. Depending on the motives, this state must not necessarily be profitable for the attacker. However, if the desired state leads to profits for the attacker, shares of this profit can be used in algorithmically enforceable side payments for the colluding players of the attack. In other words, the attacker pays to win.

It is well known in game theory that collusion and side payments pose severe challenges to mechanism design. However, the extent of the problem with regard to permissionless cryptocurrencies is not conclusively understood yet. This thesis lays the necessary groundwork for a better understanding: It describes the problem, systematizes related attacks, provides new attacks (including a proof-of-concept), evaluates the success probability and profitability, and proves that it is not possible for permissionless cryptocurrencies based on NC to prevent AIM by technical means alone. In a world where multiple cryptocurrencies co-exist and can be cryptographically interlinked, the availability of other (external) cryptocurrency resources participants care about is a plausible assumption. This requires us to reconsider the achievable economic security guarantees of permissionless cryptocurrencies and certain design decisions which amplify the problem.

Contents

Kurzfassung	ix
Abstract	xi
Contents	xiii
1 Introduction	1
1.1 Motivation	1
1.2 Basic Background	3
1.3 Problem Statement	3
1.4 Research Questions	6
1.5 Genesis and Structure	7
1.6 Main Results	10
1.7 List of Publications	12
2 Pitchforks as Offensive Protocol Updates in Nakamoto Consensus	19
2.1 Technical Background on Merged Mining	20
2.2 System Model and Goals of a Pitchfork	23
2.3 Pitchfork Attack Description	25
2.4 Countermeasures	28
2.5 Related Attacks, Potential Enhancement, and Further Analysis of Incentive Manipulation	34
3 Systematization of Algorithmic Incentive Manipulation Attacks	37
3.1 Algorithmic Incentive Manipulation	38
3.2 Generic Attack Model for AIM	39
3.3 Classification Framework for AIM	41
3.4 Example Use of Our Classification Framework	45
3.5 Classification of Existing AIM Approaches	45
3.6 Costs, Profits, and Extractable Value	53
3.7 Relation of AIM to other Attack Methods and Areas	55
4 Pay To Win Attacks	59
4.1 Relation of Pay To Win to Other AIM Attacks	60

4.2	Model	62
4.3	P2W Attack Method	63
4.4	P2W Transaction Revision, Exclusion and Ordering Attack	65
4.5	P2W Transaction Exclusion and Ordering Attack	80
4.6	Discussion and Mitigations	85
4.7	Implications of P2W and AIM	88
5	How much is the fork?	91
5.1	Calculating the Success Probability of Forks	91
5.2	Roles, Types and Sides of Players	94
5.3	Probability and Profitability Calculation	97
5.4	Implications of our Simulations	109
6	Estimating (M)EV and its Consequences	111
6.1	The History of (Miner) Extractable Value	112
6.2	Economic Rationality and Extractable Value	113
6.3	Estimation of the EEV in the Context of Attacks	119
6.4	Example Scenarios with Multiple Resources	124
6.5	Consequences of Optimizing EEV and Mitigation Strategies	130
7	Adversarial Games: Formalization of AIM attacks on Nakamoto Consensus	133
7.1	Nakamoto Consensus-based State Machine Replication	134
7.2	Types of Protocol Changes and Parallel Executions	143
7.3	Challenges of Modelling Nakamoto Consensus-based Cryptocurrencies as Games	152
7.4	Economically Rational Block Selection	159
7.5	Adversarial Game to Manipulate Transaction Ordering (no-fork)	163
7.6	Adversarial Game to Exclude Transactions (near-fork)	175
7.7	Adversarial Game to Revise Transactions (deep-fork)	182
8	Discussion and Directions for Future Work	189
8.1	Results	189
8.2	Game Theory, AIM, and Cryptocurrencies	191
8.3	Future Work	193
8.4	Conclusion	194
A	Appendices	197
A.1	Merged Mining Security Considerations	197
A.2	Optimal Transaction Ordering	198
A.3	Types of Extraction Optimizations Methods	200
A.4	Adoption of Protocol Changes and the Robin Hood Attack	203
A.5	Transaction Ordering attack (in-band)	205
A.6	Transaction Exclusion (in-band)	209

A.7 Comparison Tables for “How much is the fork”	215
A.8 Comparison of Other Definitions of Extractable Value from Related Work	220
A.9 Proof that Finite Blocks Require Protocol Updates	227
List of Figures	229
List of Tables	233
Notation, Variables and Symbols	235
Bibliography	241

Introduction

“Bitcoin works in practice, but not in theory.”

– Saying from the early days of Bitcoin also cited in [BMC⁺15]

1.1 Motivation

Since the introduction of Bitcoin [Nak08] as a model for most of today’s cryptocurrencies, this topic area has gained broad attention and some of its representatives have experienced considerable growth in terms of their exchange rate.

The technical concept that most of these systems have in common is that everybody who can contribute some physical and/or virtual resource can become part of the set of participants executing a consensus protocol. Therefore these systems are also referred to as *permissionless* [Vuk15, SJS⁺21]. In contrast, so-called *permissioned* systems require that the set of consensus participants is well defined.

A crucial part of the so-called *permissionless* Nakamoto consensus concept is the utilization of incentives in the protocol design to motivate participants (often referred to as miners or stakers) to participate in the protocol. A long-standing question in this regard is whether or not this construction is, or can be made incentive compatible, i.e., that the intended properties of a stable consensus system emerge from the appropriate (mechanism) design [BMC⁺15]. In recent years, several attacks that propose deviating strategies and potential attacks, mostly for proof-of-work systems, have been proposed (see Chapter 3 for a summary). Nevertheless, so far relatively few attacks have been observed in practice (e.g., [YSZ22, coi20, coi19b, coi19a, btg20, fom18]). On the one hand, deviating mining strategies that are theoretically more profitable than behaving honestly challenge the

incentive compatibility of Nakamoto consensus, and thus the overall stability of the consensus system. On the other hand, the lack of empirical evidence of such techniques being observed in mainstream cryptocurrencies raises the question of why they are not executed regularly in practice. Possible explanations reach from the proposed attacks not being practically feasible or stealthy enough, over missing software support and tooling, to not being profitable enough to justify potentially criminal behavior with its associated risks [Bon16, YSZ22]. This leads to the question of accurately assessing the guarantees of prevalent cryptocurrencies when considering incentive-related attacks.

This thesis aims to expose some of the problems originating from incentive manipulation attacks and their potential consequences for prevalent cryptocurrencies. We map and classify the existing attack space aimed at manipulating the incentives of actors in cryptocurrencies (see Chapter 3), improve upon known attacks (see Chapters 2 and 4) and prove the general technical feasibility of this attack category (see Chapters 4, 5 and 7). Thereby, this thesis takes a technical viewpoint and is targeted at readers with a computer science background. In spite of that, addressing the technical challenges when facilitating side payments and collusion in permissionless cryptocurrencies also relates to economic- and game theory aspects of such systems.

The problem of collusion and side payments (also referred to as bribes) between participants has long been described in game theory and is known to be the Achilles heel of mechanism design [GBI18] with results in this direction dating back to 1979 [GL79]. Especially in the early works on this subject, the problems introduced by a large number of colluding participants and side payments, was treated more like a theoretical one, since discovering and establishing the required coalitions was considered improbable [GL79]. This reasoning is based on the assessment that it is unlikely for a large number of participants to collude and conspire while at the same time being able to perform enforceable side payments among this group of apparently untrustworthy entities. It was also assumed that such a group would face societal repercussions for their (traceable) actions.

Today, this assessment is questioned by the existence of public permissionless cryptocurrencies. Essentially, such cryptocurrencies are distributed systems with a built-in payment layer operated by pseudonymous participants. Thus, automatically agreeing on coalitions among pseudonymous entities and technically enforceable payments are now in the realm of possibilities. In this thesis, we map and extend this new attack landscape and summarize it under the term algorithmic incentive manipulation (AIM). Understanding the threats arising from AIM attacks is crucial to accurately assess the security guarantees of prevalent cryptocurrency systems. As the security of most higher-level constructs, for example, *payment channels* and complex *smart contracts* also depends on the security of the underlying consensus system (often referred to as layer 1), a good understanding of the achievable security promises and guarantees in this layer is of utmost importance for all constructions that use, or are based on such permissionless cryptocurrencies.

1.2 Basic Background

The term *blockchain* is often used as an umbrella term for systems and technologies that are “kind of related to Bitcoin and cryptocurrencies”. Interestingly, this now well-known buzzword itself was not directly introduced by Satoshi Nakamoto in the original paper [Nak08], in which he was just referring to blocks and chains¹. The origins of the term can be dated back to the early Bitcoin community, where it was used to refer to specific concepts of the cryptocurrency, especially the underlying data structure [NBF⁺16]. Yet, there does not exist an agreed-upon definition of the term *blockchain*. To avoid confusion, we will use the term *distributed ledger technology* (DLT) whenever we want to refer to the broad area of technologies used in this space and the term *blockchain* for the underlying data structure.

Besides sharing similarities regarding their underlying data structure, Bitcoin and most of its cryptocurrency descendants, are based on what is termed Nakamoto consensus [BMC⁺15, SJS⁺18, DKT⁺20]. In a nutshell, proof-of-work based Nakamoto consensus (NC) enables *anyone* to initiate (valid) state transitions to a replicated state machine if they solve a cryptographic puzzle of sufficient hardness that depends on the prior state. As long as the majority of computational power continues to advance some state that is considered valid (based on predefined protocol rules), the history of past state transitions leading to the current state stabilizes over time, one step (block) at a time. This stabilization property, as well as the stabilized part of the resulting chain, is commonly referred to as *common prefix* [GKL15]. The state with the highest step count and cumulative puzzle difficulty is generally considered the currently active state of the system upon which participants build up and extend the chain to progress the system. On a technical level, this is done by appending a block of ordered transactions and an appropriate proof-of-work [DN92] to this directed, rooted, and cryptographically linked tree of other blocks. The path to the leaf with the highest depth (resp. difficulty) is called the *longest (heaviest) chain* and thus, the current state of the system.

1.3 Problem Statement

*“The incentive **may** help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. **He ought to find it more profitable to play by the rules, such rules that favour him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth.**”*

– Satoshi Nakamoto [Nak08]

(emphasis added)

¹Satoshi Nakamoto himself used the spelling *block chain* in a comment within the original source code to refer to the resulting data structure of chained hashes of blocks <https://github.com/trotskyer/original-bitcoin/blob/master/src/main.h#L795-L803>.

This original argument by Nakamoto conjectures that Bitcoin will remain stable as long as the majority of miners follow their own economic interests. Generally speaking, permissionless cryptocurrencies based on Nakamoto consensus [Nak08, SJS⁺18] (NC) require a set of miners, which makes up the majority of the voting power (e.g., hashrate), to comply and play by the rules in order to work as expected. Since Nakamoto never provided a formal description of the protocol or the incentive model behind it, this informal argument was debated within the scientific as well as the cryptocurrency research community already early on:

“In game-theoretic terms, if universal compliance were shown to be a Nash equilibrium, this would imply incentive compatibility for Bitcoin as no miner would have any incentive to unilaterally change strategy. This would imply a notion of weak stability if other equilibria exist and strong stability if universal compliance were the sole equilibrium. If on the other hand, non-compliant strategies dominate compliance, we must ask whether the resulting strategy equilibrium leads to stability for the consensus protocol.” [BMC⁺15]

Thus, a long-standing question in this context is whether such constructions are, or can be made, incentive compatible under practical conditions [BMC⁺15], i.e., the intended properties of the system emerge from the appropriate mechanism design of the protocol, which incentivizes all rational participants to act desirably. Mechanism design, as a subfield of game theory and economics, is the process of designing economic mechanisms, or incentives, to achieve a set of desired objectives in a strategic setting where players are assumed to act rationally (mainly from an economic perspective) [NR99, NR01].

To approach this question, as well as the actual security guarantees of NC, a more formal description of the underlying protocol was needed. As Nakamoto did not provide a formal description of the protocol in [Nak08], several elaborate attempts towards formalization² have been made to prove specific security properties of the protocol. Thereby, most approaches, such as [GKL15, PSS17, BMTZ17, GKR20], do assume a sufficient *honest (complacent)* majority of miners without considering incentives, or like [BGM⁺18] explicitly do not consider *bribing attacks* to manipulate the incentives of participants. Thus, in formal models of NC the question regarding incentive compatibility of NC under malicious conditions, such as bribing, was merely perceived as a niche topic.

The issue of bribing and incentive-related attacks in this context was brought forward within the academic research community by Bonneau [Bon16, Bon18]. Initially, such attacks have been deemed purely theoretical in the cryptocurrency community or highly unlikely due to their associated costs. This might be one reason why such attacks have not received broader attention right from the start. Bribing attacks target incentive compatibility and assume that at least some miners accept bribes to maximize their profit. Hereby, bribing does not necessarily refer to illegal activity but merely that a payment is made in exchange for a specific action. If the attacker, together with all bribable miners, can gain a sizable portion of the voting power (in the case of PoW computational power), even for a short period of time, attacks are likely to succeed. To the best of our knowledge,

²For a summary see [SJS⁺18].

the first discussions of *bribery* attacks on Bitcoin within the cryptocurrency community date back to a bitcointalk forum post from 2012 by a user called *cunicula* [Cun]. Since then, attacks on incentives in cryptocurrencies have been sporadically discussed in the cryptocurrency community [Ler, sAM], with the first peer-reviewed paper on the subject presented in 2016 by Bonneau [Bon16]. Over the years, several different techniques and approaches to bribing attacks have been proposed [Cun, BBH⁺13, Ler, sAM, Bon16, TJS16, LK17, LVTS17, VTL17, MHM18, JSSW18, JSZ⁺19, WHF19, KNW20, TYE21]. These proposals vary regarding their system models, technical methods, and evaluation criteria, which makes comparing them a challenging task. What all these approaches have in common is that they are targeted to manipulate the incentives of actors in the cryptocurrency ecosystem.

Also, so-called *Goldfinger* attacks [KDF13] target the incentives of participants, but here the goal of the attacker is to destroy a competing cryptocurrency to gain some undefined external utility. In [Bon18] such constructions have been revisited and deemed more relevant today than when they were initially proposed in 2013 [KDF13, BBH⁺13]. As the number of cryptocurrencies, as well as the market for trading, has evolved, shorting³ a crypto asset has become a viable strategy. The first practical example of such an attack was suggested in [Bon18] and implemented in [MHM18] as an Ethereum smart contract that rewards the mining of empty Bitcoin blocks. Other instantiations are described in [JSSW18, JSZ⁺19]. Goldfinger attacks inherently require some external utility, e.g., that the payments have to be performed in another currency unit (*out-of-band*) since, if successful, the value of the targeted cryptocurrency is intended to drop. In comparison, most classical bribing attacks have aimed at gaining *in-band* profit, i.e., in terms of cryptocurrency units of the targeted system. Therefore, they have initially been depicted as a separate form of attack compared to Goldfinger-style attacks.

Lately, also other attempts to manipulate incentives targeted to influence the *order* of transactions within a not yet mined block [EMC19, DGK⁺20, JSZ⁺19] (front-running) or to *exclude* transactions [JSZ⁺19, WHF19] have received increased attention. Hereby, the ability of a miner to select and order the transactions in his blocks has been identified as a fundamental issue of leader-based consensus protocols [KZGJ20]. It is also heavily discussed in the context of *miner extractable value* (MEV), which was initially mentioned by Daian et al. [DGK⁺20]. Although MEV was initially not well defined, it is considered to encompass additional revenue opportunities (besides block rewards and transaction fees) of miners, which originate from their ability to select and order their transactions. Front-running attacks are constantly observed in the wild already, and their exploitation, as well as prevention, is a topic of active research [DGK⁺20, fom18, ZQC⁺21, ZQT⁺20, QZG22]. The different types of ordering attacks observed in practice show that the security properties of NC under real-world incentives are still not fully understood, not only by cryptocurrency users and smart contract developers but also in the research community. In addition to this, large-scale temporary majority attacks, in which an attacker overtakes

³Short selling (shorting) refers to an investment technique in which the investor will profit if the value of the asset falls.

a cryptocurrency for a short period of time, have gained further practical importance as they have been observed more frequently in recent history [coi20, coi19b, coi19a, btg20]. This further questions the role of incentives and their targeted manipulation in the context of these systems.

All these mentioned attacks where the incentives of participants play a key role (bribing, Goldfinger, ordering), their increasing relevance in practice, as well as meta arguments [FB19, Bud18, KF19, Bon18, Bon16], have fueled the debate around incentives in Nakamoto consensus. This thesis contributes to the research in the area by addressing a couple of key questions.

1.4 Research Questions

In summary, the main research questions for this thesis are:

1.4.1 RQ1: How can bribing-, ordering-, and Goldfinger attacks on NC be systematized and better compared against each other?

Known attacks that involve the incentives of participants use different models and assumptions or have vastly different goals. Therefore, comparing them is a challenging task.

In Chapter 3 the first systematization of already proposed attacks (including the attacks developed for this thesis) is presented. The systematization allows for a better comparison of related attacks and also identifies their core characteristics. This illustrates that all these attacks are instances of the same overall category termed algorithmic incentive manipulation attacks.

1.4.2 RQ2: Is the landscape of incentive-related attacks on NC already exhaustively explored?

Given the increasing number of attacks involving incentives, a natural question is if more efficient attacks or generally other attack approaches are possible in this area. Hereby, efficiency can be defined in terms of costs or an increase in the abilities that these attacks offer.

In Chapter 2 and 4 new attacks are proposed which have new capabilities (utilize merged mining, or incentivize double-spend collusion with out-of-band payments) and are cheaper than comparable attacks described so far.

1.4.3 RQ3: How to better estimate the success probability and profitability of finite forking attacks/chain races that take the incentives of economically rational victims into account?

Prevalent attack models for chain races/forking in NC are mainly concerned with the probability calculation of infinite attacks under the assumption that the majority of

participants honestly follow the longest chain. Therefore, these models take shortcuts and ignore more complex incentives of actors, for example, their already contributed blocks or the persistence of victims to push a specific chain which is beneficial for them, instead of blindly following an attack chain as soon as it takes the lead.

In Chapter 5, techniques for probability and profitability calculation of attacks are described, which take the incentives of economically rational victims into account and, at the same time, are designed to be computationally faster than comparable approaches.

1.4.4 RQ4: How does the concept of miner extractable value (MEV) relate to incentives in NC?

The concept of MEV is considered essential in the context of ordering attacks such as front-running. As also incentives play a key role in this area, the question of how MEV relates to algorithmic incentive manipulation in NC is important for a better understanding of the associated risks.

In Chapter 6 the problem of estimating the MEV of other participants in NC-style permissionless cryptocurrencies is described, and the entire subject of ordering attacks is depicted as another instance of algorithmic incentive manipulation.

1.4.5 RQ5: Are cryptocurrencies based on NC incentive compatible such that they can be considered strongly stable?

There have been several indicators that *incentive compatibility* is too much to ask for in the context of cryptocurrencies based on NC (cf., [ES14, Bon16, Bon18, JSSW18, JSZ⁺19, FB19, KF19, JSZ⁺21b]). Although, to the best of our knowledge, this precise question has not been explicitly answered since it was proposed by Bonneau et al. [BMC⁺15].

In Chapter 8, we relate the results of this thesis to the original questions proposed by Bonneau et al. [BMC⁺15] and argue that in the light of bribing and algorithmic incentive manipulation attacks, which facilitate and finance the collusion of participants, NC cannot be considered strongly stable under *incentives other than mining income*. Especially when other resources besides the respective cryptocurrency are considered (i.e., *external-denominated utility*).

1.5 Genesis and Structure

This section chronologically reflects the genesis of this thesis. For more details regarding the methodology of the individual parts, we refer to the respective chapters of this document. At the end of this section, the overall structure of the document is presented.

1.5.1 Genesis

Originally, the scope of this thesis was more on the analysis, improvement, and application of “Bitcoin-like” blockchains in network protocols for resilient future internet applications.

Therefore, the analysis of “Bitcoin-like” protocols was one of the first expected results. As Bitcoin and thus Nakamoto consensus was presented very informally and put into operation in 2009 as a grassroots movement, not much information, as well as no detailed description of the protocol (despite the reference implementation), was available till around 2015.

As a result, the author started with a literature review and background research to correctly describe Nakamoto consensus as a distributed system and relate it to the respective literature. This led to the book *Blocks and Chains: Introduction to Bitcoin, Cryptocurrencies, and Their Consensus Mechanisms* [JSKW17]. This laid the foundation for future research in this area.

Since it was soon apparent that the resource consumption of Nakamoto consensus based on PoW, as well as the underlying performance of its prevalent implementations, is not satisfying, early research of the author focused on directions for improvement in this regard. One possible direction for improvement was merged mining, which had the potential to reduce the energy footprint of multiple cryptocurrencies while increasing the performance in terms of throughput. This is achieved by putting a hash pointer to another block header of a so-called *child* chain into the header of a PoW cryptocurrency referred to as *parent* chain. Initially, an empirical analysis of the adoption of this technique was performed. This led to the paper *Merged Mining: Curse or Cure?* [JZS⁺17], where the author analyzed various different merge mined cryptocurrencies and discovered that most of them are dominated by a subset of miners of the respective parent cryptocurrency. This paper is not directly part of this thesis, but while writing this paper the author of this thesis noticed that it is theoretically possible for mining pools to misrepresent blocks and flag them as belonging to other mining pools. This could be used in combination with merged mining to falsely blame miners in a parent cryptocurrency for “malicious” blocks while still mining regular blocks in a child cryptocurrency at the same time. The insight that such an attack, and especially merged mining, cannot easily be prevented by technical means led to the first attack described in this thesis. An extended version of this attack paper (*Pitchforks in Cryptocurrencies: Enforcing rule changes through offensive forking- and consensus techniques* [JSSW18]) is described in Chapter 2.

The theoretical possibility of this attack provided a motivating example to further investigate incentive-related attacks and thus shifted in the scope of this thesis. Therefore the author collected different incentive-related attacks on cryptocurrencies and started to systematize the scattered research on so-called *bribing*, *Goldfinger*, and *front-running* attacks. This later led to the systematization-of-knowledge paper *SoK: Algorithmic Incentive Manipulation Attacks on Permissionless PoW Cryptocurrencies* [JSZ⁺21b]. Initial versions of the SoK already identified a research gap and indicated that the attack space is not fully explored.

After the first presentation on the problem of these economic attacks, the author received good feedback and engaged in interesting discussions in the Dagstuhl Seminar 18152 *Blockchains, Smart Contracts, and Future Applications*, as well as Dagstuhl Seminar 18461 *Blockchain Security at Scale*. This led to a collaboration in identifying and

describing new algorithmic incentive manipulation attacks, which have been published in the paper *Pay To Win: Cheap, Crowdfundable, Cross-chain Algorithmic Incentive Manipulation Attacks on PoW Cryptocurrencies* [JSZ⁺21a] described in Chapter 4. As a result, the technical feasibility of new AIM attacks has been practically demonstrated, including a proof-of-concept implementation.

From the SoK [JSZ⁺21b] and the new attacks proposed in [JSZ⁺21a], it was clear to the author that this attack category could potentially cause problems for prevalent permissionless cryptocurrency designs based on Nakamoto consensus. The next logical step was to improve the analysis of attacks to more accurately assess their chance of success and, thus, their likelihood under practical conditions. The result was an improved model for analyzing finite chain races in Nakamoto consensus, which was presented in the paper *How Much is the Fork? Fast Probability and Profitability Calculation during Temporary Forks* [JSSW22b] is described in Chapter 5 of this thesis.

The question of how attacks of miners could be discovered in practice was addressed in preliminary empirical research to better identify forks [SSJ⁺19] as well as the internal structure of mining pools [RJZH19]. Both works are not directly part of this thesis.

In parallel, the term miner extractable value (MEV) was coined by Daian et al. [DGK⁺20] and led to a series of research papers regarding front-running, arbitrage, and the exploitation of transaction ordering in the context of single leader protocols such as Nakamoto consensus. As there is a strong relation between bribing and MEV, the paper *Estimating (Miner) Extractable Value is Hard, Let's Go Shopping!* [JSSW22a] was written to highlight that a bribe can be viewed as another source of MEV. This relates the topic of extractable value to the problem of algorithmic incentive manipulation in Nakamoto consensus protocols.

Through the proof-of-concept, in [JSZ⁺21a] (and Chapter 4), we have shown that certain AIM attacks are technically feasible. The remaining question is, do these attacks only work in this particular instance (for precisely these cryptocurrencies), or can such attacks be generalized to any Nakamoto consensus-based cryptocurrency system? To investigate under which assumptions and exact conditions such attacks are feasible, Chapter 7 formalizes AIM attacks on Nakamoto consensus by constructing them as adversarial games exposing their core design properties. Therefore, a practically oriented model of Nakamoto consensus, using elements of state machine replication (SMR), has been created as a prerequisite. Under this model, the technical possibility of constructing adversarial games to perform algorithmic incentive manipulation is shown.

1.5.2 High-level Structure

Chapter 2 provides a motivating example to further investigate incentive-related attacks by describing such an attack on Nakamoto consensus-style permissionless cryptocurrencies based on proof-of-work. The described attack utilizes merged mining and proves that in this setting, the problem is not easily mitigated, e.g., through counter-attacks.

Chapter 3 systematizes known attacks that are designed to programmatically manipulate the incentives of actors within a cryptocurrency system and summarizes them under the term *algorithmic incentive manipulation* (AIM). Thereby, they are classified according to their capabilities and intended consequences. This highlights that the attack space has not been exhaustively explored.

As a consequence, a new attack method called pay to win (P2W) has been proposed in Chapter 4, including a proof-of-concept implementation of the attack targeting Bitcoin with the help of Ethereum smart contracts.

To better estimate the success probability and profitability of attacks resulting in chain races in Nakamoto consensus, Chapter 5 describes a model which takes the incentives of rational victims of such attacks into account. The model also accounts for other real-world conditions, such as time constraints and already contributed blocks.

Chapter 6 discusses the relation of algorithmic incentive manipulation (AIM) to the concept of miner extractable value (MEV). Thereby, bribes are depicted as just another source of extractable value - illustrating the problem of meaningfully bounding MEV if bribes are issued in other cryptocurrency systems.

To generalize the PoC presented in Chapter 4, Chapter 7 aims to provide a unified view of algorithmic incentive manipulation (AIM) attacks. To explicitly highlight and describe the requirements, necessary assumptions, and core design properties, Chapter 7 formalizes AIM attacks on Nakamoto consensus by constructing them as adversarial games. Therefore, a state machine replication (SMR) based model of Nakamoto consensus is described.

Chapter 8 discusses the results of the thesis and relates them to the area of game theory. At the end, interesting areas of future research are outlined.

1.6 Main Results

In summary, the main results from this thesis are:

- A new attack called *Pitchfork* (see Chapter 2) demonstrates that public permissionless PoW-based cryptocurrencies are vulnerable to the (malicious) interlinking of their protocol rules using merged mining. Moreover, we provide proof that such an attack can only be counter-attacked if the attacker hashrate is below $1/3$, or below $1/4$ depending on the side-effects of the direct counter-attack [JSSW18] (see Section 2.4.2).
- The first systematization of bribing, Goldfinger, Pitchfork, front-running, and other related attacks are described as different instances of algorithmic incentive manipulation (AIM) attacks [JSZ⁺21b] (see Chapter 3). At the same time, we also provide a comprehensive classification of different attacks according to properties,

such as their impact on the consensus layer of cryptocurrencies (i.e., no-fork, near-fork, deep-fork).

- A description of new AIM attacks called *Pay-to-win* (P2W) that is capable to revert, reorder or exclude any transaction from a targeted public permissionless cryptocurrency based on NC, provided that the attacker has sufficient budget on a different funding cryptocurrency [JSZ⁺21a] (see Chapter 4). Our Analysis includes simulations as well as a proof-of-concept implementation of the attack.
 - To the best of our knowledge, the analysis of our P2W attacks also includes the first analysis of finite chain races, where the overall duration of the attack is the main limiting factor (see Section 4.4.3), as well as the first analysis regarding the synchrony between two chains (see Section 4.4.4).
 - All calculations, source code, and artefacts are publicly available on Github⁴.
- An improved model to calculate the success probability as well as the profitability of finite AIM attacks [JSSW22b] (see Chapter 5).
 - To the best of our knowledge, this is the first model which takes already contributed blocks, as well as the incentives of economically rational victims, into account.
 - In the case of forks, where the attacker is one block behind and tries to catch-up infinitely long, his profitability on the fork increases beyond his expected profit for staying on the original chain when the miner has a hashrate greater than $\approx 38.2\%$ of the total hashrate. This value already occurred in previous publications in the context of cryptocurrencies [TJS16, MHM18] but was not discussed in great detail there. We show how this value is derived and where it comes from and prove that it is precisely $1/\phi^2$ in Section 5.3.1.
 - All calculations, source code, and artefacts are publicly available on Github⁵.
- A relation between *miner extractable value* (MEV) and AIM, showing that the MEV of other participants cannot readily be estimated, especially if they are interested in more than one resource/cryptocurrency [JSSW22a] (see Chapter 6).
 - All calculations and artefacts are publicly available on Github⁶.
- A model of Nakamoto consensus-based state machine replication (SMR) was designed (see Chapter 7). Using this model, it was possible to show that adversarial games attacking the mechanism design of permissionless cryptocurrencies based on NC can technically always be constructed, provided that the targeted system is efficiently verifiable and eventually makes progress. Using the described techniques, an attacker can incentivize desired orderings, the exclusion of transactions, or

⁴https://github.com/kernoelpanic/pay2win_artefacts

⁵https://github.com/kernoelpanic/howmuchisthefork_artefacts

⁶https://github.com/kernoelpanic/estimatingMEVishard_artefacts

deep-forks (see Section 7.5, 7.6 and 7.7). Moreover, it was possible to better describe and compare different types of temporary and permanent forking events utilizing the newly designed model. To the best of our knowledge, this is the first formal description of these previously not well-defined forking events, commonly referred to as *soft-forks* and *hard-forks*, as well as their associated rule changes (see Section 7.2).

1.7 List of Publications

This thesis is based on a series of publications co-authored by the author of the thesis. Hereby, all except two publications are reused almost verbatim in this thesis in the individual chapters, which represent partially revised versions of those publications with small changes mainly to improve spelling and clarity. All papers used in their entirety are exclusively used in this thesis and have not been used in any other thesis. From the paper *A Wild Velvet Fork Appears!* [ZSJ⁺18] only a small text snippet relating the terms hard-fork and soft-fork to the cryptocurrency literature has been used verbatim. Beyond that, only the initial concept for the table was reused, but in a corrected, formalized, and extended version. In all used papers, except the latter, the author of this thesis has been the main author and lead.

Chapter 2 is an extended version of the paper *Pitchforks in Cryptocurrencies* [JSSW18].

The following is the list of publications (in chronological order) on which this thesis is based on. For each publication, the chapter is given in which content from this publication is used.

- Estimating (Miner) Extractable Value is Hard, Let's Go Shopping!
 - **Bibentry:** Aljosha Judmayer, Nicholas Stifter, Philipp Schindler, and Edgar Weippl. Estimating (Miner) Extractable Value is Hard, Let's Go Shopping! In *Financial Cryptography and Data Security. International Workshops - CoDecFin*, 2022
 - **Artefacts:** All source code for simulations, calculations and figures available on github⁷.
 - **Eprint:** <https://eprint.iacr.org/2022/359.pdf>
 - **Chapter:** Chapter 6 is based on this publication.
- How much is the fork? Fast Probability and Profitability Calculation during Temporary Forks
 - **Bibentry:** Aljosha Judmayer, Nicholas Stifter, Philipp Schindler, and Edgar Weippl. How much is the fork? Fast Probability and Profitability

⁷https://github.com/kernoelpanic/estimatingMEVishard_artefacts

- Calculation during Temporary Forks. In *1st International Cryptoasset Analytics Workshop (CAAW) co-located with the International World Wide Web Conference (WWW) 2022*, 2022
- **Artefacts:** Implementation of the model, all source code used for simulations, calculations, figures available on github⁸.
 - **Eprint:** <https://eprint.iacr.org/2021/1231.pdf>
 - **Chapter:** Chapter 5 is based on this publication.
- Pay To Win: Cheap, Crowdfundable, Cross-chain Algorithmic Incentive Manipulation Attacks on PoW Cryptocurrencies
 - **Bibentry:** **Aljosha Judmayer**, Nicholas Stifter, Alexei Zamyatin, Itay Tsabary, Ittay Eyal, Peter Gazi, Sarah Meiklejohn, and Edgar R. Weippl. Pay to Win: Cheap, Cross-Chain Bribing Attacks on PoW Cryptocurrencies. In Matthew Bernhard, Andrea Bracciali, Lewis Gudgeon, Thomas Haines, Ariah Klages-Mundt, Shin’ichiro Matsuo, Daniel Perez, Massimiliano Sala, and Sam Werner, editors, *Financial Cryptography and Data Security. FC 2021 International Workshops - CoDecFin, DeFi, VOTING, and WTSC, Virtual Event, March 5, 2021, Revised Selected Papers*, volume 12676 of *Lecture Notes in Computer Science*, pages 533–549. Springer, 2021
 - **Artefacts:** Prototype implementation, all source code used for simulations, calculations, figures and data evaluation available on github⁹. There also exists a video of the conference talk.
 - **Eprint:** <https://eprint.iacr.org/2019/775.pdf>
 - **Chapter:** Chapter 4 is based on this publication.
 - SoK: Algorithmic Incentive Manipulation Attacks on Permissionless PoW Cryptocurrencies
 - **Bibentry:** **Aljosha Judmayer**, Nicholas Stifter, Alexei Zamyatin, Itay Tsabary, Ittay Eyal, Peter Gazi, Sarah Meiklejohn, and Edgar R. Weippl. SoK: Algorithmic Incentive Manipulation Attacks on Permissionless PoW Cryptocurrencies. In Matthew Bernhard, Andrea Bracciali, Lewis Gudgeon, Thomas Haines, Ariah Klages-Mundt, Shin’ichiro Matsuo, Daniel Perez, Massimiliano Sala, and Sam Werner, editors, *Financial Cryptography and Data Security. FC 2021 International Workshops - CoDecFin, DeFi, VOTING, and WTSC, Virtual Event, March 5, 2021, Revised Selected Papers*, volume 12676 of *Lecture Notes in Computer Science*, pages 507–532. Springer, 2021
 - **Artefacts:** Comparison table of different algorithmic incentive manipulation attacks as included in this thesis. There also exists a video of the conference talk.

⁸https://github.com/kernoelpanic/howmuchisthefork_artefacts

⁹https://github.com/kernoelpanic/pay2win_artefacts

- **Eprint:** <https://eprint.iacr.org/2020/1614.pdf>
- **Chapter:** Chapter 3 is based on this publication.
- Pitchforks in Cryptocurrencies: Enforcing rule changes through offensive forking- and consensus techniques (Short Paper)
 - **Bibentry:** **Aljosha Judmayer**, Nicholas Stifter, Philipp Schindler, and Edgar R. Weippl. Pitchforks in Cryptocurrencies: Enforcing Rule Changes Through Offensive Forking- and Consensus Techniques (Short Paper). In Joaquín García-Alfaro, Jordi Herrera-Joancomartí, Giovanni Livraga, and Ruben Rios, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2018 International Workshops, DPM 2018 and CBT 2018, Barcelona, Spain, September 6-7, 2018, Proceedings*, volume 11025 of *Lecture Notes in Computer Science*, pages 197–206. Springer, 2018
 - **Artefacts:** Calculation of attack bounds for hashrate and visualization.
 - **Eprint:** <https://eprint.iacr.org/2018/836.pdf>
 - **Chapter:** Chapter 2 is based on this publication.
- A Wild Velvet Fork Appears! Inclusive Blockchain Protocol Changes in Practice (Short Paper)
 - **Bibentry:** Alexei Zamyatin, Nicholas Stifter, **Aljosha Judmayer**, Philipp Schindler, Edgar Weippl, and William J. Knottebelt. A Wild Velvet Fork Appears! Inclusive Blockchain Protocol Changes in Practice (Short). In *5th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security 18 (FC). Springer*, 2018. (Short Paper)
 - **Artefacts:** Table for classifying different fork types as included in this thesis.
 - **Eprint:** <https://eprint.iacr.org/2018/087.pdf>
 - **Chaptere:** Some text snippets relating the terms hard-fork and soft-fork to the cryptocurrency literature have been reused in Chapter 7, .

1.7.1 List of Publications not covered in this Thesis

The following is a list of publications (in chronological order) that are *not* covered in this thesis directly, but to which the author of this thesis contributed while pursuing his PhD.

- Opportunistic Algorithmic Double-Spending: How I learned to stop worrying and love the Fork
Nicholas Stifter, **Aljosha Judmayer**, Philipp Schindler, and Edgar Weippl. Opportunistic Algorithmic Double-Spending: How I learned to stop worrying and hedge the Fork. In *ESORICS, 2022*
- RandRunner: Distributed Randomness from Trapdoor VDFs with Strong Uniqueness

- Philipp Schindler, **Aljosha Judmayer**, Markus Hittmeir, Nicholas Stifter, and Edgar R. Weippl. RandRunner: Distributed Randomness from Trapdoor VDFs with Strong Uniqueness. In *Network and Distributed System Security Symposium 2021*, February 2021
- Unnecessary Input Heuristics and PayJoin Transactions

Simin Ghesmati, Andreas Kern, **Aljosha Judmayer**, Nicholas Stifter, and Edgar R. Weippl. Unnecessary Input Heuristics and PayJoin Transactions. In *HCI International 2021 - Posters*, July 2021
 - HydRand: Practical Continuous Distributed Randomness

Philipp Schindler, **Aljosha Judmayer**, Nicholas Stifter, and Edgar Weippl. HydRand: Practical Continuous Distributed Randomness. In *Proceedings of IEEE Symposium on Security and Privacy (IEEE S&P)*. IEEE, 2020
 - Blockchain-Technologie – Anwendungsformen

Nicholas Stifter, Philipp Schindler, and **Aljosha Judmayer**. Blockchain-Technologie – Anwendungsformen. Technical report, April 2021. Published: LexisNexis
 - Blockchain-Technologie – technische Erklärung

Aljosha Judmayer, Philipp Schindler, and Nicholas Stifter. Blockchain-Technologie – Technische Erklärung. Technical report, January 2020. Published: LexisNexis
 - A Deep Dive into Bitcoin Mining Pools: An Empirical Analysis of Mining Shares

Matteo Romiti, **Aljosha Judmayer**, Alexei Zamyatin, and Bernhard Haslhofer. A Deep Dive into Bitcoin Mining Pools: An Empirical Analysis of Mining Shares. In *The 2019 Workshop on the Economics of Information Security*, 2019
 - Echoes of the Past: Recovering Blockchain Metrics From Merged Mining

Nicholas Stifter, Philipp Schindler, **Aljosha Judmayer**, Alexei Zamyatin, Andreas Kern, and Edgar Weippl. Echoes of the Past: Recovering Blockchain Metrics From Merged Mining. In *Proceedings of the 23rd International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2019
 - Revisiting Practical Byzantine Fault Tolerance Through Blockchain Technologies

Nicholas Stifter, Aljosha Aljosha, and Edgar R. Weippl. Revisiting Practical Byzantine Fault Tolerance Through Blockchain Technologies. In Stefan Biffl, Matthias Eckhart, Arndt Lüder, and Edgar R. Weippl, editors, *Security and Quality in Cyber-Physical Systems Engineering, With Forewords by Robert M. Lee and Tom Gilb*, pages 471–495. Springer, 2019
 - Proof-of-Blackouts? How Proof-of-Work Cryptocurrencies Could Affect Power Grids

Johanna Ullrich, Nicholas Stifter, **Aljosha Judmayer**, Adrian Dabrowski, and Edgar Weippl. Proof-of-Blackouts? How Proof-of-Work Cryptocurrencies Could Affect Power Grids. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 184–203. Springer, 2018
 - Merged Mining: Curse or Cure?

Aljosha Judmayer, Alexei Zamyatin, Nicholas Stifter, Artemios G. Voyiatzis, and Edgar R. Weippl. Merged Mining: Curse or Cure? In Joaquín García-Alfaro, Guillermo Navarro-Arribas, Hannes Hartenstein, and Jordi Herrera-Joancomartí, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2017 International Workshops, DPM 2017 and CBT 2017, Oslo, Norway, September 14-15, 2017, Proceedings*, volume 10436 of *Lecture Notes in Computer Science*, pages 316–333. Springer, 2017

- **A Holistic Approach to Smart Contract Security**
Nicholas Stifter, **Aljosha Judmayer**, and Edgar R. Weippl. A Holistic Approach to Smart Contract Security. *ERCIM News*, 2017(110), 2017
- **Bitcoin - Cryptocurrencies and Alternative Applications**
Aljosha Judmayer, Alexei Zamyatin, Nicholas Stifter, and Edgar R. Weippl. Bitcoin - Cryptocurrencies and Alternative Applications. *ERCIM News*, 2017(110), 2017
- **A performance assessment of network address shuffling in IoT systems (Extended Abstract)**
Georg Merzdovnik, **Aljosha Judmayer**, Artemios G. Voyiatzis, and Edgar Weippl. A performance assessment of network address shuffling in IoT systems (Extended Abstract). 2017
- **Lightweight Address Hopping for Defending the IPv6 IoT**
Aljosha Judmayer, Johanna Ullrich, Georg Merzdovnik, Artemios G. Voyiatzis, and Edgar R. Weippl. Lightweight Address Hopping for Defending the IPv6 IoT. In *12th International Conference on Availability, Reliability and Security (ARES)*, 2017
- **Blocks and Chains: Introduction to Bitcoin, Cryptocurrencies, and Their Consensus Mechanisms**
Aljosha Judmayer, Nicholas Stifter, Katharina Krombholz, and Edgar Weippl. Blocks and Chains: Introduction to Bitcoin, Cryptocurrencies, and Their Consensus Mechanisms. *Synthesis Lectures on Information Security, Privacy, and Trust*, 9(1):1–123, 2017
- **The Other Side of the Coin: User Experiences with Bitcoin Security and Privacy**
Katharina Krombholz, **Aljosha Judmayer**, Matthias Gusenbauer, and Edgar R. Weippl. The Other Side of the Coin: User Experiences with Bitcoin Security and Privacy. In *International Conference on Financial Cryptography and Data Security (FC)*, 2016
- **Tutorials on Cryptocurrencies**
Aljosha Judmayer and Edgar R. Weippl. Cryptographic Currencies Crash Course (C4): Tutorial. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11-15, 2016, Companion Volume*, pages 1021–1024, 2016
Aljosha Judmayer and Edgar R. Weippl. Condensed Cryptographic Currencies Crash Course (C5): Tutorial. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1857–1858, 2016
- **On the security of security extensions for IP-based KNX networks**

Aljosha Judmayer, Lukas Krammer, and Wolfgang Kastner. On the security of security extensions for IP-based KNX networks. In Maria Gradinariu Potop-Butucaru and Hervé Rivano, editors, *10th IEEE International Workshop on Factory Communication Systems (WFCS 2014)*, Toulouse, France, May 2014

Pitchforks as Offensive Protocol Updates in Nakamoto Consensus

“There is nothing permanent except change.”

– Attributed to Heraclitus

In this chapter¹, a new incentive manipulation attack is proposed which utilizes out-of-band payments and merged mining to subsidize the attack. Our attack illustrates that certain attacks on cryptocurrencies cannot trivially be counter-attacked even if the “honest” nodes control more than 50% of the total hashrate. This should provide a motivating example to analyse the concept of incentive related attacks on cryptocurrencies in greater detail.

Forking in general is a core design property of Nakamoto consensus (NC). Even without the interference of malicious actors, the proposal of a block by two different miners at approximately the same point in time can trigger a fork. Such a fork eventually resolves when new blocks are appended on top of either branch. Apart from these temporary forks that are resolved when one branch takes the lead, there also exists the concept of a permanent fork, also referred to as a *chain split*. In such a case, the end result are two different systems. This can happen intentionally or unintentionally, for example, when the underlying protocol has to be updated as in case of the Bitcoin LevelDB upgrade².

In this context, the loosely defined terms *hard-fork* and *soft-fork* have been in use as descriptors of different classes of upgrade mechanisms for the underlying rules. For a formal definition of different forking methods we refer to Section 7.2. For now we stick to the informal definitions used within the cryptocurrency community. Within the

¹This chapter represents an extended version of publication [JSSW18].

²Cf. <https://github.com/bitcoin/bips/blob/master/bip-0050.mediawiki>

cryptocurrency community, the term *hard-fork* has established itself [BMC⁺15, GCR16] as a descriptor for protocol changes that can incur a permanent split of the blockchain, as they permit or even enforce the creation of blocks that are considered invalid under previous protocol rules. In contrast to this, *soft-forks* in principle allow to retain compatibility with older protocol versions, specifically towards clients adhering to previous protocol rules. Apart from these two more frequently used types of forks, also an approach termed *velvet fork*, which expands upon the concept of a soft-fork, was outlined by Kiayias et al. [KMZ17]. Specifically, velvet forks intend to avoid the possibility of disagreement by a change of rules through rendering modifications to the protocol backward compatible *and* inclusive to legacy blocks. So, in contrast to temporary forks, which can happen during normal operation, hard-, soft- and velvet forks refer to a protocol upgrade that might or might not lead to a permanent fork, i.e., a chain split.

We expand upon the concept of different protocol changes by discussing the potential security implications arising from permanent hard-forks (protocol changes) that are interleaved with the original protocol/system in a malicious way. We describe the *pitchfork* as an example to enforce rule changes in a targeted cryptocurrency, denoted as Π . In a pitchfork, the attacker creates a different protocol version Π' and uses techniques from merged mining to interleave his new hard-forked cryptocurrency with the targeted parent chain Π s.t., mining “bad blocks” in the parent chain is a prerequisite for being accepted as a valid PoW in the new cryptocurrency which consists of the pitchfork child chain Π' .

Our construction highlights some interesting questions. In particular, with regards to the underlying (game-theoretic) incentive model, such attacks can lead to negative side effects in permissionless cryptocurrencies based on NC.

2.1 Technical Background on Merged Mining

Merged mining was originally conceived as a bootstrap technique [JZS⁺17], aiming to increase the PoW difficulty and, as a consequence, the security of other cryptocurrencies in their early stage, when they are more vulnerable to dishonest miners. Merged mining aims to improve blockchain security by rapidly increasing the number of nodes participating in the distributed consensus. The key idea of merged mining is to allow a *child* blockchain (e.g., Namecoin) to accept valid PoW produced for another *parent* blockchain (e.g., Bitcoin), provided that they meet the hardness criteria of the receiving (child) blockchain even if they do not meet the criteria of the sending (parent) blockchain.

Merged mining was first implemented in Namecoin. By accepting Bitcoin blocks through merged mining, Namecoin quickly achieved a high difficulty level. Other popular altcoins, including Litecoin and Dogecoin, have also adopted merged mining, helping to establish it as a *de facto* hardening mechanism for PoW altcoins.

The implementation of merged mining has not been without controversy. There are

already discussions on realistic threats on network centralization³ and scam attacks⁴. Unfortunately, the technique of merged mining has never been formally described or well documented. Thus we provide an informal description of the process on the example of Bitcoin. Our description is based one of the few “official” sources on the subject, namely the Bitcoin wiki [Namb].

In brief, merged mining can be described as a technique to solve and disseminate PoW puzzles for more than one PoW blockchain simultaneously, while the computational effort for solving the PoW puzzle is performed only once. In this sense, the *parent* blockchain miners are solving the puzzles and the *child* or *auxiliary* blockchains adapt their operation so as to accept these solved PoW puzzles.

To participate in merged mining, a miner must connect (using a full node) to each child blockchain network in which she wants to participate in. This is required to collect and construct valid blocks for the respective child blockchain. Of course, all the child chains have to accept merge mined blocks from a selected parent cryptocurrency as valid, given that the difficulty or other requirements are met. The respective parent blockchain must support a method to link to, or include, some arbitrary data in its block headers. For a merged mined block to be valid, this selected data field must contain a cryptographic link (e.g., a hash) to a valid block of the respective child blockchain. For the case of Bitcoin as a parent blockchain, this requirement is fulfilled by using the structure of the *coinbase transaction*, depicted in Figure 2.1.

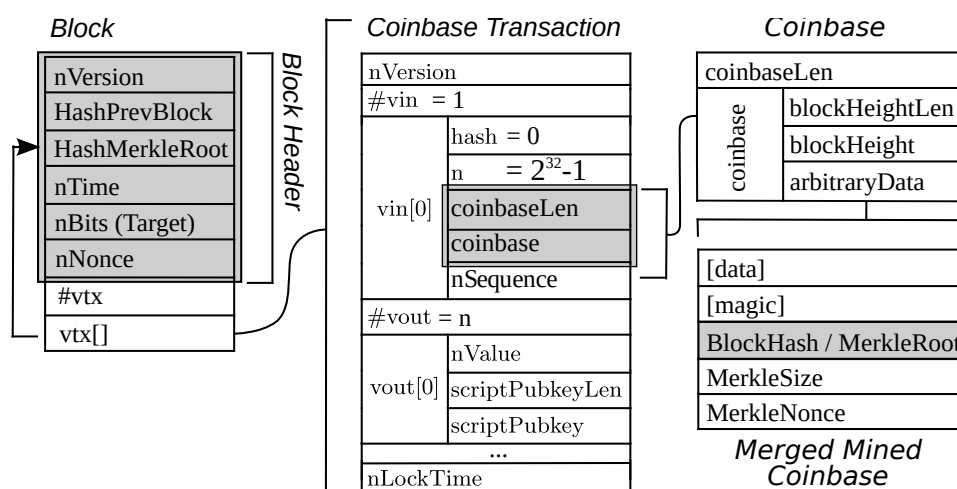


Figure 2.1: Data structures and location of the hash placed in a Bitcoin block when Bitcoin is used as a parent chain in the context of merged mining. The *BlockHash / MerkleRoot* is pointing to the respective block(s) in the merge mined child chain(s).

³Cf. <https://www.cryptocompare.com/mining/guides/what-is-merged-mining-bitcoin-namecoin-litecoin-dogecoin/>

⁴Cf. commentary on the Eligius CoiledCoin scam, available on <https://bitcointalk.org/index.php?topic=56675.msg678006#msg678006>

The coinbase transaction is a special type of transaction for rewarding the block miner. It comprises n *transaction outputs* (denoted as `vout[0]`), which transfers the mined coins to the account(s) of the miners, and one special *transaction input* (denoted as `vin[0]`). This special input includes the “block reward” (denoted as `nValue`) and the “coinbase” fields. The *block reward* comprises new cryptocurrency units that are created up until the maximum supply is reached. The *coinbase* encodes the current *block height* and can contain up to 96 bytes of arbitrary data, as summarized in Table 2.1.

Field name		Type (Size)	Description
coinbaseLen		VarInt (1-9 bytes)	Length of the coinbase field in bytes as a variable length integer. Maximum size is 100 bytes.
coinb.	blockHeightLen	(1 bytes)	Length in bytes required to represent the current <code>blockHeight</code> .
	blockHeight	(3 bytes)	Current block height.
	[data]	char[] (0-52 bytes)	Optional: Arbitrary data that can be filled by the miner (e.g., identifying the block miner)
	[magic]	char[] (4 bytes)	Optional: If $len(coinbase) \geq 20$, <code>magic</code> bytes indicate the start of the merged mining information, e.g., “\xfa\xbe”
	BlockHash or MerkleRoot	char[] (32 bytes)	Hash of the merge-mined block header. If more than one cryptocurrency is merge-mined, this is the Merkle tree root hash of those cryptocurrencies.
	MerkleSize	uint32_t (4 bytes)	Size of the Merkle tree, i.e., the maximum number of contained cryptocurrencies.
MerkleNonce	uint32_t (4 bytes)	Used to calculate the indices of the mined cryptocurrencies in the Merkle tree. If no Merkle tree is used, it is set to 0.	

Table 2.1: Structure of the coinbase of a merge-mined block. Uses Namecoin as an example [Namb]

In the context of merged mining, the last 40 bytes of the `coinbase` field can be used to store information for the child blockchain. If merged mining involves only one child blockchain, then 32 bytes define a `BlockHash`, i.e., the hash of the block header of the child blockchain directly. If more than one child blockchains are involved, the 32 bytes form the `MerkleRoot`, i.e., the root hash or a Merkle tree of size `MerkleSize`. The leaves of the tree represent the hashes of the block header of each child blockchain.

It is vital to ensure that merged mining does not occur for multiple forks of the same child blockchain; this would compromise the security of the latter. This is addressed as follows: Each child blockchain has a fixed `chainID` that is defined by its developers and hard-coded in its client implementation. For example, the `chainID` for Namecoin is set⁵ to the value `0x0001`. Every miner can choose freely for how many and for which

⁵Cf. Namecoin source code on <https://github.com/namecoin/namecoin-core/blob/fdfb20fc263a72acc2a3c460b56b64245c1bedcb/src/chainparams.cpp#L123>

PoW child blockchains they want to perform merged mining and, hence, maintain a different Merkle tree. The combination of `MerkleSize`, `MerkleNonce`, and `chainID` are fed to a linear congruential generator so as to produce the unique position of a child blockchain `chainID` on a Merkle tree of a given size⁶.

For more merged mining related security considerations, see Appendix A.1.

2.2 System Model and Goals of a Pitchfork

The increasing number of cryptocurrencies, as well as the rising number of actors within every single cryptocurrency, inevitably leads to tensions between the respective communities. As with open-source projects, (protocol) forks could be the result of broad disagreement. After a permanent fork (chain split), both communities “mine” their own business and the conflict can be considered resolved from a technological perspective. This has been observed for example with Bitcoin and its various derivatives such as Bitcoin Cash or Bitcoin Gold. But what if this is not the case? In this chapter, we outline the possibility of malicious forking and consensus techniques that aim at destroying the other branch of a protocol hard-fork (protocol update). Thereby, we illustrate how merged mining can be used as an attack method against a permissionless PoW cryptocurrency, which itself involuntarily serves as the parent chain for an attacking merge mined branch of a hard-fork.

Merged mining is already known for posing a potential issue to the child cryptocurrencies, for example demonstrated in the case of CoiledCoin⁷. However, so far, no concrete example has been given that merged mining can also pose a risk to the parent chain. Since (parent) cryptocurrencies cannot easily prevent being merge mined⁸, an attack strategy using this approach would be applicable against a variety of permissionless PoW cryptocurrencies. In this chapter, we describe a scenario where merged mining is used as a form of attack against a parent chain in the context of a hostile protocol fork.

2.2.1 Actors

For our attack scenario, we assume a permissionless PoW-based cryptocurrency II, whose miners cannot agree on whether or not to change the consensus rules. Some of the miners want to adapt the consensus rules in a way such that newly mined blocks may not be valid under the old rules, i.e., perform a hard-fork, or protocol update. Thereby, we differentiate between the following actors:

⁶Cf. Namecoin source code on <https://github.com/namecoin/namecoin-core/blob/dfb20fc263a72acc2a3c460b56b64245c1bedcb/src/auxpow.cpp#L177-L200>

⁷Cf. <https://bitcointalk.org/index.php?topic=56675.msg678006#msg678006>

⁸The inclusion of a hash value within a block to provably attributed it to the creator of the proof-of-work (PoW) is enough to support merged mining [JZS⁺17]

- **Antiquated or backward-compatible miners (\mathcal{A}):** The fraction of miners (with hashrate $p_{\mathcal{A}}$) in a currently active cryptocurrency Π that does not want to change the consensus rules of Π .
- **Byzantine, or change enforcing miners (\mathcal{B}):** The fraction of miners (with hashrate $p_{\mathcal{B}}$) in a currently active cryptocurrency Π , that wants to change the consensus rules, i.e., perform a hard-fork/protocol update. Moreover, they want that only their branch of the fork survives.
- **Indifferent, independent, or neutral miners (\mathcal{I}):** The set of miners (with hashrate $p_{\mathcal{I}}$) that has no strong opinion on whether or not to change the consensus rules. They want to maximize their profits and act economically rational to achieve this goal, with the limitation that they want to avoid changes as far as possible. If there is no imminent need that justifies the implementation costs for adapting to changes, they will not react⁹.

For our example, we assume that \mathcal{B} wants to increase the block size, while \mathcal{A} does not want to implement any rule change. The goal of the attackers in \mathcal{B} is twofold: 1) *Enforce* a change of the consensus rules in the target cryptocurrency. 2) *Disrupt* the operation of the old branch of the hard-forked target cryptocurrency (Π), which does not follow the new consensus rules Π' .

2.2.2 Characteristics of a Pitchfork

For this chapter, we are only interested in forking scenarios that are *not bilateral*. In a bilateral fork, conflicting changes are intentionally introduced to ensure that two separate cryptocurrencies emerge [ZSJ⁺18]. An example of such a scenario would be the changed chain ID between Ethereum and Ethereum Classic. It is commonly believed that in a non-bilateral forking event, the only reliable possibility to enforce a change requires that the majority of the mining power supports the change. Thereby, two main cases can be distinguished according to [ZSJ⁺18], namely a *reducing* change as well as an *expanding* change. For more detail we refer to Section 7.2 of this thesis.

If the introduced change *reduces* the number of blocks that are considered valid under the new consensus rules, all new blocks mined according to the new rules are still considered valid under the old rules, but some blocks which previously would have been considered valid are no longer considered valid under the new rules. An example of such a scenario would be a *block size decrease*. In this case, the first goal (*enforce*) of our attack is easy to achieve if $p_{\mathcal{B}} > p_{\mathcal{A}} + p_{\mathcal{I}}$ holds since any fork introduced by $p_{\mathcal{B}}$ will eventually become the longest chain and be adopted by $p_{\mathcal{A}}$ and $p_{\mathcal{I}}$ because of the longest heaviest chain rule. Therefore, in this case, the rules have been successfully changed from Π to a subset Π' since $p_{\mathcal{A}}$ is the minority in this case. If \mathcal{A} decides to continue a cryptocurrency under

⁹This should capture the observation that not all miners immediately perform merged mining if it is possible, even though it would be rational to do so [JZS⁺17].

the original rules Π , such that larger blocks are again possible, they have to declare themselves as a new currency since the original one has been overtaken and the rules changed to a subset. Therefore, the goal to *enforce* is clearly reached in such a case. However, the *disruption* goal cannot be reached directly if all miners in \mathcal{A} create a new cryptocurrency that follows the original rules Π .

If the introduced change *expands* the set of blocks that are considered valid under the new consensus rules, then some blocks following the new rules will not be considered valid under the old rules. Therefore, any mined block that is only valid under the new rules will cause a fork. An example of such a scenario would be a *block size increase*¹⁰. In this case, a permanent hard-fork will only occur if the chain containing blocks following the new rules grows faster, i.e., $p_{\mathcal{B}} > p_{\mathcal{A}} + p_{\mathcal{I}}$ holds. The result would be that the forking event creates two different currencies: cryptocurrency Π' , which includes big blocks, and cryptocurrency Π , which forked from the main chain after the first big block. Therefore, again the *disruption* goal cannot be reached directly. To reach this goal, some miners in \mathcal{B} could be required to switch to the original cryptocurrency Π and disrupt its regular operation, e.g., by mining empty blocks. This, of course, has the drawback that the respective attacking miners that switched from Π' to Π do not gain any profits in Π' , and their rewards in Π will be worthless if they succeed in rendering Π unusable.

The *pitchfork* attack method proposed in this chapter aims to achieve both attack goals simultaneously, even in cases where $p_{\mathcal{B}} < p_{\mathcal{A}} + p_{\mathcal{I}}$ holds.

2.3 Pitchfork Attack Description

The basic idea of a pitchfork attack is to use merged mining as a form of attack against the other branch of a fork in a permissionless PoW cryptocurrency resulting from a disputed consensus rule change. The pitchfork should disrupt the normal operation of the attacked branch to such an extent that the miners abandon the attacked branch and switch to the branch of the hard-fork, which performs merged mining and follows the new consensus rules. We call the cryptocurrency up to the point of the fork *ancestor* cryptocurrency $\bar{\Pi}$. After the forking event, the cryptocurrency which still follows the same rules is denoted as Π , whereas the *change enforcing* cryptocurrency branch that uses merged mining and the new consensus rules is denoted as Π' .

To execute the attack, the new merge mined branch Π' accepts valid *empty blocks* of Π as a PoW for Π' . In the nomenclature of merged mining, the chain Π , which should be attacked, is called the *parent chain*, and chain Π' is called the *child chain*. For a valid parent block b of Π , the following additional requirements need to be satisfied: i) The block b has to be empty. Therefore, the contained Merkle tree root in the header of the respective block must only include the hash of the (mandatory) coinbase transaction. Given the corresponding coinbase transaction, it can then be verified that b is indeed

¹⁰Our example, in which \mathcal{B} wants to increase the block size and \mathcal{A} does not want to implement any rule change, would resemble such an *expanding* protocol change.

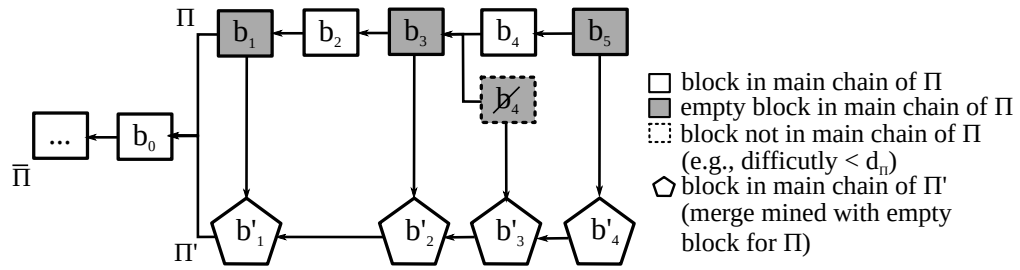


Figure 2.2: Example of blocks mined after the hard-fork into two cryptocurrencies Π and Π' .

empty. ii) The coinbase transaction of b must include the hash of a valid block b' for Π' . The header of block b' contains a Merkle tree root with the actual transactions performed in Π' .

Figure 2.2 shows the two cryptocurrencies after the fork. The last block in the ancestor cryptocurrency $\bar{\Pi}$ before the forking event is b_0 . The first empty block that is merge mined is b_1 in this example. This block (b_1) is valid under the old rules and fulfills the difficulty target in Π , except that it is empty. Moreover, the block b_1 was mined by a miner in \mathcal{B} , which happens with probability p_B , and contains the hash of block b'_1 in its coinbase. Therefore b_1 serves as a valid PoW for Π' as well. Block b_2 was not mined by a miner in \mathcal{B} , which happens with probability $1 - (p_B + p_T)$, and therefore it is not empty and does not contain a hash for a valid block for Π' in its coinbase. This shows that the two chains are not necessarily synchronized regarding their number of blocks. The block interval in Π' depends on the difficulty target of Π' . Since we assume that the attacker does not control the majority of the hashrate ($p_B < p_A + p_T$), the difficulty D in Π' should be lower than in Π at the beginning of the attack, i.e., $D_{\Pi'} < D_{\Pi}$ holds. If the difficulty has been adjusted in Π' , then the overall number of blocks should be approximately the same for both chains. In such a case, there might be empty blocks such as b_4 , which do fulfill the difficulty target for Π' but not for Π . Still, if $D_{\Pi'} < D_{\Pi}$ holds, then over time, a fraction of all blocks in Π corresponding to p_B will be mined by a miner in \mathcal{B} . If we assume that $p_B \approx 0.33$, then approximately every third block in Π should be empty.

PoW Difficulty: Theoretically, it would be possible that Π' requires the same or even a higher difficulty than Π . If $D_{\Pi'} \geq D_{\Pi}$, then chain Π' would contain fewer blocks than chain Π . This, of course, would have a negative effect on the latency in chain Π' , i.e., the time it takes till a transaction is confirmed. However, any merge mined blocks that meet the difficulty requirement of $D_{\Pi'}$ will be considered valid in Π . For example, when $D_{\Pi'} = D_{\Pi}$, the number of blocks in Π' relative to Π would only correspond to the fraction of the hashrate (p_B) that performs merged mining. Nevertheless, since chain Π' increased the block size, the throughput could theoretically remain the same or even be higher than in chain Π (depending on the actual implementation). Some examples

regarding an increased block size are discussed in [GKW⁺16, CDE⁺16]. Alternatively, Bitcoin-NG [EGSvR16] could also be applicable. The latter approach would have the added benefit that the negative impact on latency and confirmation times is mitigated. To illustrate our attack, it is not of particular relevance which adaptation is used to increase the throughput in Π' .

2.3.1 Effects of the Attack

In the simplest case, if no countermeasures are taken by the chain under attack, a pitchfork reduces the throughput of the target chain Π by the number of empty blocks corresponding to the hashrate of the attackers ($p_{\mathcal{B}}$). Considering the limited block size in Π and events in Bitcoin, or other cryptocurrencies, where the number of unconfirmed transactions in the mempool peaked, a hashrate of $p_{\mathcal{B}} \approx 0.33$ mining solely empty blocks, would likely have an impact on the duration of such periods of congestion, and hence also transaction fees and confirmation times. This could sway both users and miners in \mathcal{I} to switch to the attacking chain Π' , which further reinforces the attack. Two other advantages of the attack are that it is *pseudonymous* and that the risk in terms of currency units in Π , as well as its severeness, is *parameterizable*.

Pseudonymous: Since the pitchfork attack is executed by miners through producing new blocks that are, in addition, merged mined with the attacking chain, it is in theory possible to hide the identities of the attackers because no unspent transaction outputs need to be involved in the attack that could have a traceable history. However, additional care needs to be taken by these miners to ensure that their identity is not inadvertently revealed through their behavior [JZS⁺17].

Parameterizable: The attack is not an all-in-move, and its costs, in terms of currency units in Π , can be parameterized. The goal of the attack is to disrupt the original chain Π , but if this fails, the attackers may not lose much. Due to merged mining, the main costs of a failed attack result from the foregone profits from transaction fees that are not collected in chain Π . Additional costs created by merged mining, i.e., running an additional full node for chain Π can be negligible compared to the overall costs related to mining [NBF⁺16]. Moreover, even a failed attack on Π can still be profitable for the attacking miners since the attackers in \mathcal{B} are early adopters of Π' . If the value of the newly created cryptocurrency Π' increases enough, the additional income may not only compensate for the reduced income from mining empty blocks in Π but could even create a surplus for the miners in Π' . In addition, the attack can be made compatible with other available cryptocurrencies that can be merged mined with Π . Therefore, additional revenue channels from existing merge mined cryptocurrencies are not affected by the pitchfork and can even help to subsidize the attack.

As a further parameterization for the attack, it is also possible to execute it in stages. To test whether there is enough support for chain Π' , it is possible to first start with a relatively low risk to the attackers by not requiring them to mine empty blocks and instead

only demand the creation of smaller blocks that can still include high fee transactions. From there, the attackers can reduce the number of permissible transactions step by step. At a final stage, all coins earned through mining empty blocks in Π can also be used to fund additional attacks, such as triggering additional spam transactions in Π as soon as the cooldown period of 100 blocks has passed. For instance, splitting the coinbase rewards into many individual outputs of a high enough value with different lock times and rendering the output scripts as *anyone can spend* can lead to a large influx of additional transactions as users (and miners) compete to scoop up these free currency units. This is easy to verify as an additional rule in Π' . However, more complex attack scenarios such as those outlined in [Bon16, TJS16, LK17] may also be included as additional consensus rules.

2.4 Countermeasures

In this section, we outline some countermeasures that can be taken by players in \mathcal{A} , as well as their effectiveness.

2.4.1 Exclude Empty Blocks in Π

The miners in \mathcal{A} can decide to fork off empty blocks and just build on top of blocks containing transactions. This requires the coordinated action of all miners in \mathcal{A} . If $p_{\mathcal{A}} > p_{\mathcal{B}} + p_{\mathcal{I}}$, this approach will work in general. A possible counter-reaction by the attackers in Π' would be to introduce dummy transactions to themselves in their blocks in Π . Therefore, it has to be ensured that those transactions are indeed dummies. For example: All used output addresses of every transaction belong to the same entity, but this must not be possible to correlate given just the block b_n in Π . One way to achieve this is to require that all output addresses in a block have been derived from the miner's public-key of the respective block, like in a Hierarchically Deterministic (HD) Wallet¹¹ construction. The *master public-key property* of such a construction allows that future ECDSA public-keys can be derived from current ones. This is done by adding a multiplication of the base point with a scalar value to the current public-key. The corresponding secret-key is derived in the same manner but can only be computed by its owner. If it is not possible to perform a transaction to an address for which the miner does not have the corresponding private key, the utility of a block only containing transactions of the respective miner is very limited for users of Π . To check this condition on an arbitrary block b_n , the public-key of the miner, as well as the scalar value for the multiplication, is required. These values can be added to the coinbase transaction of the corresponding block b'_n in Π' .

If such dummy transactions are used, the miners of \mathcal{A} would be required to monitor the chain of Π' to deduce which block in Π has been merged mined with Π' and includes only dummy transactions. If the miners of \mathcal{A} find such a block, they can still cause a

¹¹Cf. BIP32 <https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki>

fork in Π to ignore it. Besides being more complex, this also poses a potential risk for all transactions in Π . Since the block b_n could be released before b'_n , there is no way to tell whether or not b_n was indeed merged mined. And hence includes a hash to b'_n before b'_n has been published in Π' . With this knowledge, miners in \mathcal{B} can intentionally create forks in Π by holding back new blocks in Π' for a while. By slightly relaxing the rules for dummy transactions and allowing, for example, one transaction output address that is not required to be derivable by the HD construction, double-spends can be executed more easily in Π . In this particular case, miners of merged mined blocks can include a regular transaction that they want to double-spend in their block, being assured that this block will get excluded in retrospect by all miners $p_{\mathcal{A}}$ in Π if b'_n is released in Π' . Therefore, more fine-grained exclusion rules on the transaction level would be necessary.

These examples illustrate that it is non-trivial to change the consensus rules in Π such that the effects of a pitchfork attack are mitigated. Every change of the defenders in \mathcal{A} leads to an arms race with the attackers in \mathcal{B} . Moreover, excluding all merge mined blocks in Π requires active monitoring of Π' to detect them. Therefore, at least the miners in \mathcal{A} have to change their individual consensus rules for Π – which they wanted to avoid in the first place.

2.4.2 Launch a Counter-attack on Π'

Miners in \mathcal{A} can use their mining power to counter-attack the attacking chain Π' . However, this has several limitations: Since every block in Π' requires an empty parent block in Π as part of its PoW, miners cannot create empty merge mined blocks in Π' while at the same time creating full blocks in Π . To stall the pitchfork Π' , at least a fraction of $p_{\mathcal{A}}$ ($p'_{\mathcal{A}} \leq p_{\mathcal{A}}$) has to mine empty blocks in Π to also create empty merge mined blocks for Π' . Though, thereby the counter-attackers could actually help the pitchfork attack.

To clearly overtake the pitchfork chain Π' , the counter-attacking miners need to have more than 50% of the hashrate in Π' . If not, the lost throughput caused by empty blocks in Π' might be compensated by the increased block size. This introduces the first constraint for the counter-attackers, i.e., that the hashrate $p'_{\mathcal{A}}$ they dedicate to the counter-attack must be larger than the attack hashrate $p'_{\mathcal{A}} > p_{\mathcal{B}}$.

However, the counter-attackers must also take care not to push the total hashrate $p'_{\mathcal{A}} + p_{\mathcal{B}}$, which is dedicated towards attacking Π over 50%. Otherwise, more destructive attack rules than mining empty blocks may be rendered effective. For example, requiring non-empty blocks to be ignored or anyone-can-spend transactions. If the defenders retain the majority in Π , and if they are able to reliably identify all merge-mined attack blocks, they can exclude them in Π . Thus, the second constraint requires that for a counter-attack, the bound $p_{\mathcal{B}} + p'_{\mathcal{A}} < 0.5$ for the share of blocks in the heaviest chain of Π holds.

Depending on the exact implementation of merged mining in Π' , the counter-attackers have some options to avoid that their empty blocks for Π , which they are required to provide as PoW for Π' , cause further harm to Π . Therefore, we differentiate between

counter-attacks without direct negative consequences on Π and *counter-attacks with direct negative consequences* on Π .

An example for a counter-attack without direct negative consequences on Π would be to only submit PoW solutions to Π' that fulfill the difficulty target for Π' but not for Π . This counter-attack approach has the marked disadvantage that any block meeting the difficulty target of Π also cannot be submitted as solutions in Π' , effectively reducing the counter-attackers' hashrate p'_A in Π' by a factor dependent on the particular difference in difficulty between Π' and Π . A better counter-attack without direct negative consequences on Π can be achieved if the defenders intentionally construct blocks for the parent chain Π that are unlikely to end up in the main chain yet are still accepted as a valid proof-of-work in Π' . For instance, stale branches in Π could be created and extended. However, this is only effective if the freshness requirements for parent blocks in Π' are not too tight. In both cases, since p'_A is no longer contributing toward the effective hashrate of Π , its remaining honest miners, $p_I + p_A - p'_A$ must still retain a hashrate that exceeds that of the adversary to ensure that honest blocks constitute a majority of the heaviest chain. Therefore, the original attacker gains an advantage from merged mining since he can use his full hashrate in both chains at the same time. Moreover, the counter-attacking fraction of the miners would forgo their rewards in Π for the duration of the counter-attack.

We now compare the two cases of counter-attacks with and without direct negative consequences on the parent chain Π and calculate the maximum tolerable hashrate of p_B , such that the counter-attack succeeds in dominating at least one of the two systems, i.e., \mathcal{A} has more than half of the hashrate on at least one system. For this analysis, we make the simplifying assumption that the total hashrate of indifferent miners p_I is zero. Hence the total hashrate of \mathcal{A} is $p_A = 1 - p_B$. This hashrate can be split between the two chains Π and Π' arbitrarily to launch the counter-attack.

Counter-attack without Direct Negative Consequences on Π

In this case, we assume that direct negative consequences from the pitchfork attack on the parent chain Π can be avoided while creating merge mined child chain blocks for Π . For example, merged mined empty blocks of the parent chain are accepted as a valid PoW for the child chain, even if they do not have a valid predecessor in the parent chain. Indirect negative consequences of the counter-attack, like, for example, an overall hashrate reduction that works on the longest heaviest chain in Π , is ignored for this analysis.

Figure 2.3 shows the hashrates achievable by \mathcal{A} on the respective chains Π and Π' for defending and counter-attacking. The figure is parameterized by different values for the hashrate of the pitchfork attacker (p_B). It can be observed that in this case, an attacker with $p_B > \frac{1}{3}$ total hashrate cannot be countered on both chains simultaneously without losing the majority $p_A < 0.5$ on one of the two chains.

Theorem 1. Assuming the invested counter-attack hashrate does *not* directly strengthen the attack on the merge mined parent chain Π . Then, to successfully perform a counter-

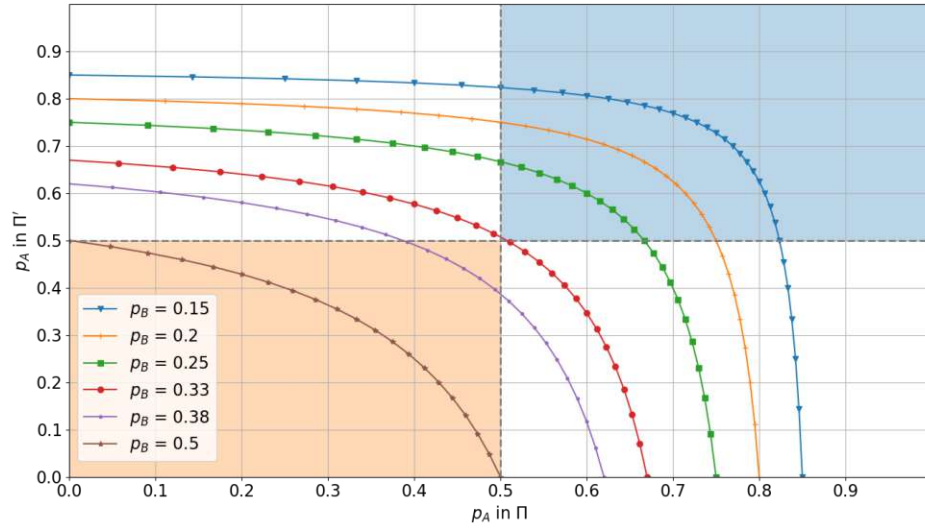


Figure 2.3: Hashrates of the defender/counter-attacker in the respective chains Π and Π' for different values of pitchfork attacker hashrate p_B . The colored orange square marks the area in which the counter-attackers \mathcal{A} loses in both chains, whereas the colored blue square marks the area in which the counter-attackers \mathcal{A} can retain the majority in both systems.

attack in which \mathcal{A} dominates a merge minded pitchfork child chain Π' , the attacker hashrate p_B must be less than $\frac{1}{3}$, such that \mathcal{A} can dominate both chains (i.e., have more than half of the hashrate in both chains).

Proof. Since the pitchfork attack utilizes merged mining, the hashrate of the attacker p_B is available in both systems, Π and Π' . To overtake and dominate the merge mined child chain Π' , the most a defender with hashrate p_A can invest in terms of hashrate is $\frac{1}{2}$, as otherwise, we would lose in protocol Π . To keep control of the parent, the following inequality has to hold: $\frac{p_A}{2} > p_B$. Since we are in a two-player model, the hashrates are defined as: $p_A = 1 - p_B$. Replacing p_A in our inequality then gives us the maximum hashrate of the attacker $\frac{1-p_B}{2} > p_B$, which simplifies to $p_B < \frac{1}{3}$, and provides the upper bound for the hashrate of the attacker s.t., the defender \mathcal{A} can win in both chains. \square

Figure 2.4 shows a visualization of this scenario in which merged mining the parent is possible without causing the negative side effects intended by the pitchfork attack on the parent chain. In this example, $p_B = \frac{1}{3}$ and thus no party can clearly win either chain.

This result for permissionless PoW cryptocurrencies has an interesting relation to a paper by Lindell et al. [LLR04], in which the authors prove that authenticated Byzantine

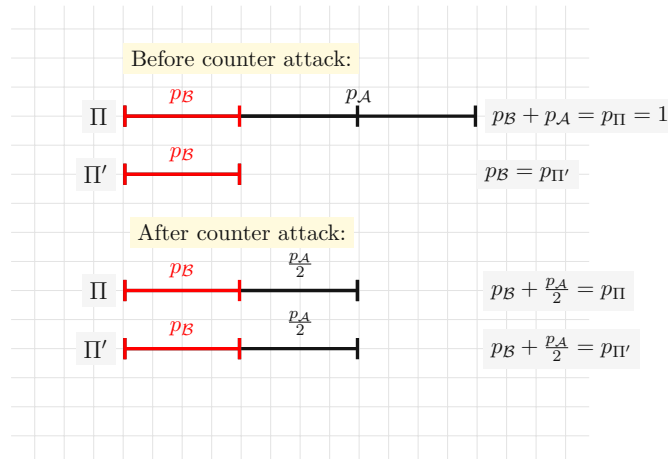


Figure 2.4: Visual comparison of hashrates in the two systems Π and Π' , before and after counter attacking the merge minded pitchfork protocol Π' , when merged mining of the parent is possible without negative effects on the parent chain. In this example, the hashrate of the attacker is $p_B = \frac{1}{3}$, which results in a situation where no player can clearly win on any of the two chains in case of a counter-attack.

Agreement protocols only remain secure under parallel or concurrent composition (even for just two executions), when more than $2/3$ of the participating parties are honest.

Counter-attack with Direct Negative Consequences on Π

In this case, we assume that direct negative consequences of the attack on the parent chain Π cannot be avoided while creating merge mined child chain blocks for Π .

Theorem 2. Assuming the invested counter-attack hashrate directly strengthens the attack on the merge mined parent chain Π . Then, to successfully perform a counter-attack in which \mathcal{A} dominates a merge mined pitchfork child chain Π' , with no ability to avoid damage on the target chain, the attacker hashrate p_B must be less than $\frac{1}{4}$, s.t. \mathcal{A} can dominate both chains (i.e., have more than half of the hashrate on both chains).

Proof. Since the pitchfork attack utilizes merged mining, the hashrate of the attacker is available in both systems. The difference now is that the defenders can not switch parts of their hashrate to the pitchfork chain running Π' without harming the target chain running Π . We denote the hashrate that switches and gets “malicious” in Π with p'_A . To determine the maximum p_B , the following conditions have to hold:

$$p_B + p'_A < p_A \quad (\text{Ensures dominance of } \mathcal{A} \text{ on targeted parent chain}) \quad (2.1)$$

$$p_B < p'_A \quad (\text{Ensures dominance of } \mathcal{A} \text{ on child chain}) \quad (2.2)$$

$$p_B + p'_A + p_A = 1 \quad (\text{Overall hashrate is bound by 1}) \quad (2.3)$$

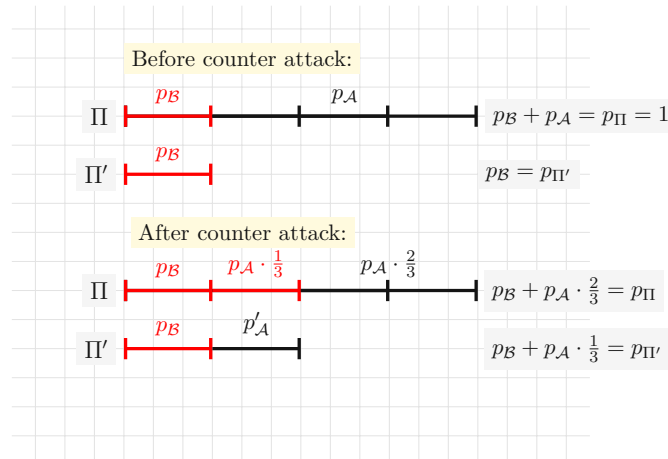


Figure 2.5: Visual comparison of hashrates in the two systems Π and Π' , before and after counter attacking the merge minded pitchfork protocol Π' , when merged mining of the parent is not possible without negative side effects on the parent chain. In this example, the hashrate of the attacker is $p_B = \frac{1}{4}$, which results in a situation where no player can clearly win on any of the two chains in case of a counter-attack.

This gives us $p_A > \frac{1}{2}$ and $p_B < \frac{1-p_A}{2}$. Solving this for the point where the hashrate/power on both chains is equal (i.e., $p_B = p'_A$ and $p_B + p'_A = p_A$) gives us the upper bound $p_B < \frac{1}{4}$ for the attacker hashrate s.t., the defender \mathcal{A} can win on both chains.

$$p_B + p'_A - p_A = 0 \quad (2.4)$$

$$p_B + p'_A + p_A = 1 \quad (2.5)$$

$$p_B = p'_A \quad (2.6)$$

$$2p_B + 2p_B = 1 \quad (2.7)$$

$$p_B = \frac{1}{4} \quad (2.8)$$

□

Figure 2.5 shows a visualization of this scenario in which merged mining the parent is not possible without negative side effects on the parent chain. In this example, $p_B = \frac{1}{4}$ and thus no party can clearly win either chain in case of a counter-attack.

2.4.3 Do nothing

Given the undesired side effects of the other countermeasures discussed so far, one option for the defenders \mathcal{A} would be to do nothing. Then, the users of Π have to live with a reduced throughput in the size of p_B . This would work and dry out the funds of the attackers \mathcal{B} , if the reduced gains from missed transaction fees in Π cannot be

compensated by the new gains of mining an additional cryptocurrency, Π' . If the gains in Π' overcompensate the losses in Π , then the surplus in funds can be reinvested into new mining hardware to increase the overall share in the mining ecosystem until p_B accounts for the majority of mining power. Then, they would be able to overtake Π directly.

In other words, the success of ignoring the pitchfork attack depends on the balance between lost income and newly gained income. If the loss in income from pitchfork mining Π can be avoided, for example, if the consensus rules of Π are designed in a way such that the transaction fees in Π do not count towards the income of the miner but instead are burned, then mining empty blocks would not have any disadvantage. Another option would be to require more complex attacks on Π as a PoW for Π' . Such attacks might even be profitable in the target cryptocurrency Π .

2.5 Related Attacks, Potential Enhancement, and Further Analysis of Incentive Manipulation

In [JZS⁺17], it is argued that merged mining could also be used as an attack vector against the parent chain. However, no concrete examples are given. In this chapter, we outline that merged mining can be used as an attack method against a PoW cryptocurrency in the context of a hostile protocol hard-fork. The pitchfork attack is an example of such a technique, which poses interesting questions regarding the interplay of different cryptocurrencies, as well as their incentive structures. The difficulties regarding counter-attacking a Pitchfork attack on a permissionless PoW cryptocurrency have an interesting relation to a paper by Lindell et al. [LLR04], in which the authors prove that authenticated Byzantine Agreement protocols only remain secure under parallel or concurrent composition (even for just two executions) when more than $2/3$ of the participating parties are honest. In the case of classical Byzantine fault tolerant systems, this boils down to the requirement that there has to exist at least one honest node which must be aware of a concurrent alternative partition of nodes before making his decision to support only one partition. In other words, this honest node is the tie-breaker and cannot be tricked to also support an alternative conflicting agreement in which dishonest nodes have equivocated i.e., participate in a violation of the safety property. This leads to the well known $2f + 1$ bound i.e., of more than $2/3$ honest nodes. Compared to the Pitchfork attack, merged mining can be viewed as the “equivocation” and the requirement that the defenders need to “win” ($> 51\%$) at least one of the two systems leads to the bound for the adversarial hashrate that supports each concurrent system of less than $1/3$ i.e., more than $2/3$ honest hashrate.

A natural question now is if there are other attack possibilities on permissionless PoW cryptocurrencies leveraging such incentive and concurrency issues. Different bribing methods that can be used in hostile blockchain takeovers are described in [Bon18], placing the focus on attacks where the attacker has an extrinsic motivation to disrupt the consensus process, i.e., Goldfinger attacks [KDF13]. The example given in this chapter is a concrete instance of such a situation. The pitchfork attack can also be viewed as a

2.5. Related Attacks, Potential Enhancement, and Further Analysis of Incentive Manipulation

subsidized Goldfinger attack. Therefore, some of the described methods for Goldfinger attacks might also be used in conjunction with our proposed attack. This also holds true for the large body of work on bribing [Bon16, MHM18] and incentive attacks that distract the hashrate of participants [TJS16, VTL17].

The general idea of such an *offensive consensus attack*, like the pitchfork, is that the participants of the offensive system are required to provably attack a different system as part of the consensus rules. We show that such attacks are theoretically possible and can lead to an arms race in which defenders are forced to adapt their consensus rules. Still, the consequences, as well as the economic and game-theoretic incentives of such attacks, have yet to be analyzed in greater detail to better understand if they are practicable and, if so, how to protect against them. Therefore, we first systematize the related work on other incentive attacks in the next chapter of this thesis.

Systematization of Algorithmic Incentive Manipulation Attacks

This chapter¹ aims to systematize the landscape of research on attacks that target the incentives of actors within – and through the use of – cryptocurrencies. Bribing and other related attacks aim to directly influence the incentives of actors within (or outside) the targeted cryptocurrency system. The theoretical feasibility of bribing attacks on cryptocurrencies was highlighted in the academic community in 2016 by Bonneau [Bon16], with various different techniques and approaches having since been proposed. Some of these attacks are designed to gain in-band profits in terms of currency units, while others intend to break the mechanism design and render the targeted cryptocurrency worthless. Recent reports of real-world 51% attacks on smaller cryptocurrencies [coi20, coi19b, coi19a, btg20] underline the realistic threat of large-scale attacks on permissionless cryptocurrencies. Moreover, they show that incentives play a vital role in this regard. In this chapter, we systematically expose the extensive but scattered body of research in this area that has accumulated over the years. We summarize these bribing attacks and similar techniques that leverage programmatic execution and verification under the term *algorithmic incentive manipulation* (AIM) attacks, as they all intend to influence the incentives of actors. Thereby, we show that the problem space is not yet fully explored. We present several research gaps and opportunities that warrant further investigation based on our analysis. In particular, we highlight *no- and near-fork* attacks as a powerful, yet underestimated, AIM category that raises security concerns not only for smart contract platforms. To succeed, such attacks require forks of short length independent of a security parameter k defined by the victim, or even no-forks at all. The consequences, such as *transaction exclusion and ordering* manipulation, raise security concerns for certain established use cases of smart contract platforms.

¹This chapter represents an extended version of publication [JSZ⁺21b].

To systematically expose the large body of research on bribing-, front-running- Goldfinger- and other related attacks, we jointly consider them under the general term *algorithmic incentive manipulation attacks* (AIM). Thereby, we want to distinguish programmatic ways to set up and execute incentive attacks on cryptocurrencies by using cryptocurrencies from “classical” bribing attacks, for example, using a suitcase full of cash to bribe miners. The difference is that the latter does not require technical means but, at the same time, lacks technical enforcement [Bon16].

The classification of AIM attacks in this chapter forms a prerequisite and basis for comparing and discussing different attacks in this area, as well as meta arguments [FB19]. In summary, the contributions of this chapter are:

1. A definition and discussion of algorithmic incentive manipulation (AIM) providing a unified view of different approaches targeting the incentives of actors in cryptocurrencies.
2. A generic attack model for AIM suitable to compare most of the available attacks. This attack model will be further extended throughout the thesis.
3. A classification framework for AIM that is applicable to describe a broad class of attacks.
4. A classification of existing AIM approaches and discussion of main observations and gaps regarding their capabilities.

3.1 Algorithmic Incentive Manipulation

To meaningfully partake in a Nakamoto consensus protocol, a certain *capacity* [Bon18] of a scarce resource is required. In the case of Proof-of-Work (PoW), these resources are mining hardware and electricity to solve cryptographic puzzles. In [Bon18], the different ways to gain capacity in Nakamoto consensus are grouped into four different strategies: *rent*, *bribe*, *build* and *buy out*. It is well known that an actor who builds a new datacenter running specialized mining hardware, rents GPU clusters, or buys existing mining hardware from current miners can increase his influence on the targeted cryptocurrency and thereby (depending on his resources) potentially launch attacks [Bon18, Bud18]. The desired property of Nakamoto consensus (based on PoW²) of being *permissionless* [Vuk15] allows such attacks.

In this thesis, we want to focus on *algorithmic incentive manipulation* (AIM) methods to gain capacity in permissionless PoW-based cryptocurrencies, as all existing attacks which fall into this category – and are classified in this chapter – explicitly target PoW systems. Nevertheless, the basic principles of AIM should also apply to most PoS cryptocurrencies. Algorithmic (i.e., purely “virtual”) methods of gaining capacity rely on the usage of game

²In comparison, in proof-of-stake (PoS) cryptocurrencies it would not be possible to rent or build *new* capacity, as all stake eligible for voting has to exist in the system already [Bon18].

theory and cryptocurrencies to perform cryptographically guaranteed payments that are dependent on certain conditions. This ability of cryptocurrencies to tie payments to the fulfillment of certain conditions, for example, the existence of prior transactions, or the successful execution of smart contract invocations, is a way to promise *conditioned* payments.

AIM methods do not involve the physical transfer of resources, like buying or building, and maintaining hardware. Instead, these methods assume that at least some fraction of actors, within or outside of the system, behaves rationally in the sense that they want to maximize their profits. Some approaches for AIM have been referred to as *bribing*, but AIM goes beyond what is currently viewed as a bribing attack in the literature, as they should incorporate Goldfinger [KDF13] and front-running [EMC19, DGK⁺20] attacks as well. Therefore our broader definition is as follows:

Definition 1. Algorithmic Incentive Manipulation (AIM) works by posing credible threats, or promising conditioned rewards (denominated in cryptocurrency units), in an algorithmically enforceable way. Thereby, AIM incentivizes certain actions within a targeted cryptocurrency system.

3.2 Generic Attack Model for AIM

For all analyzed AIM attacks in this chapter, we describe the following generic attack model, which can readily be applied in most cases.³ If an analyzed attack deviates from this model, it is highlighted in detail when the attack is described. This encompasses additional assumptions and augmentations relevant to specific attacks.

As most bribing and related attacks in this area are designed to target Bitcoin, Ethereum, or other derived cryptocurrencies, we also focus in our model on AIM in *permissionless* [Vuk15] proof-of-work (PoW) cryptocurrencies. That is, we assume protocols adhering to the design principles of Bitcoin, more formally described in subsequent works [Nak08, GKL16, PSS17] and sometimes referred to as Nakamoto consensus [BMC⁺15, SJS⁺18, DKT⁺20]. Within the attacked cryptocurrency, we differentiate between *miners*, who participate in the consensus protocol and attempt to solve PoW-puzzles, and *clients*, who do not engage in such activities. As in previous work on bribing attacks [LK17, TJS16, MHM18, Bon16], the set of miners is assumed to be fixed, as well as their respective computational power within the network is assumed to remain constant.

To abstract from currency details, we use the term *value* as a universal denomination for the purchasing power of a certain amount of cryptocurrency units or any other out-of-band funds such as fiat currency. Miners and clients may own cryptocurrency units and are able

³Only the Proof-of-Stake blocks [LVTS17, VTL17] attack, as well as Fomo 3D [fom18], are fundamentally different: The former is targeted to attack mining pools, while the latter is designed as an exit scam, but can also lead to scenarios resembling a censorship attack.

to transfer them (i.e., their value) by creating and broadcasting valid transactions within the network. Moreover, like in most prior work [TE18, LK17, MHM18], the simplifying assumption is made that exchange rates remain constant throughout the attack if we are not dealing with Goldfinger-style attacks that specifically aim to harm the exchange rate.

3.2.1 Actors

Our generalized attack model splits participating miners into three groups whose roles remain static for the attack duration. Categories follow the *BAR* (*Byzantine, Altruistic, Rational*) [AAC⁺05, LCW⁺06] behavior model. Additionally, we define the *victim(s)* as another group or individual without hashrate.

- **Byzantine miners:** The set of byzantine miners, also referred to as attacker(s), \mathcal{B} wants to execute an AIM attack on a *target cryptocurrency*. Therefore, \mathcal{B} is in control of some bribing funds $\text{FUNDS}(\mathcal{B}) > 0$ that can either be in-band (i.e., in currency units of the target cryptocurrency) or out-of-band (e.g., in some other cryptocurrency), depending on the attack scenario. Furthermore, \mathcal{B} has some or no hashrate $p_{\mathcal{B}} \geq 0$ in the target cryptocurrency. The attacker(s) may deviate arbitrarily from the protocol rules.
- **Altruistic or honest miners:** Altruistic miners \mathcal{A} , are honest and always follow the protocol rules, hence they will not accept bribes to mine on a different chain-state or deviate from the rules, even if it would offer a larger profit. Altruistic miners \mathcal{A} control some hashrate $p_{\mathcal{A}}$. If there are none in the target cryptocurrency, they are modeled by having 0 hashrate, thus $p_{\mathcal{A}} \geq 0$.
- **Rational or bribable miners:** Rational miners \mathcal{R} control hashrate $p_{\mathcal{R}} > 0$ in the target cryptocurrency. They aim to maximize their profits in terms of *value* and act economically rational to achieve this goal. We consider such miners “bribable”, i.e., they follow strategies that deviate from the protocol rules as long as they are expected to yield higher profits than being honest. For our analyses, we assume that rational miners do not have the opportunity to concurrently engage in other rational strategies such as selfish mining [ES14].
- **Victim(s):** The set of victims \mathcal{V} or a single victim, which loses value if the bribing attack is to be successful. The victims control zero hashrate and can thus be viewed as a client. They, might control additional funds which they can use for counter-attacks.

It holds that $p_{\mathcal{B}} + p_{\mathcal{A}} + p_{\mathcal{R}} = 1$. The assumption that the victim of an AIM attack has no hashrate is plausible, as the majority of transactions in Bitcoin or Ethereum are made by clients who do not have any hashrate in the system they are using. Nonetheless, this assumption is often left implicit (e.g., [LK17]).

Some bribing attacks (e.g., [TJS16]) implicitly model victims (in this case, the betrayed collaborators of the double-spending attack) as honest, i.e., as strictly following the protocol. We emphasize that this is not necessarily the case, especially if economically rational counter-attacks by the victim should be considered. This distinction between rational and honest victims is more important if \mathcal{V} directly controls some hashrate, but even in a setting where \mathcal{V} has no hashrate, he can use his funds ($\text{FUNDS}(\mathcal{V})$) for counter-attacks⁴. Although, we would also follow the argument in [Bon16] that requiring clients to perform counter-bribing is undesirable.

Whenever we refer to an attack as *trustless*, we imply that no centralized trusted third party is needed between the briber and the bribee to ensure correct payments are performed for the desired actions. It is desirable from the attacker’s perspective to design AIM attacks so that the attacker(s), as well as the collaborating miners, have no incentive to betray each other if they are economically rational.

3.2.2 Communication and Timing

As previous AIM attacks, we assume that all miners in the target cryptocurrency have *perfect knowledge* about the attack once it has started, if not stated differently. Miners with imperfect information about an ongoing AIM attack can be modeled by adding their respective hashrate to the hashrate of altruistic miners ($p_{\mathcal{A}}$) as they would continue their regular mining activity. All participants communicate through message passing over a peer-to-peer gossip network, which we assume implements a reliable broadcast functionality. This does not mean that every transmitted transaction will make it into the next block, as the block size is bounded by the underlying blockchain protocol. Analogous to [GKL16], we model the adversary as “rushing”, meaning that he gets to see all other players’ messages before deciding his strategy.

If more than one cryptocurrency is involved in the considered scenario, for example, when out-of-band payments should be performed in another cryptocurrency, an additional *funding cryptocurrency* is assumed. While the attack is performed on a *target cryptocurrency*, the funding cryptocurrency is used to orchestrate and fund it. In such a case, we also assume that the difficulty and thus the mean block interval of the funding chain is fixed for the duration of the attack. Further, no additional attacks are concurrently being launched against either of the cryptocurrencies.

3.3 Classification Framework for AIM

We first introduce a general classification along four main dimensions: the *state of the targeted transaction(s)*; the *intended impact* on these transactions; the *required interference*

⁴Therefore in Section 3.7 we distinguish between rational and honest/altruistic victims to allow for a more fine-grained discussion of the presented attacks.

with *consensus*, i.e., the depth of blockchain reorganizations caused by forks for the attack to be successful; and finally the *used payment methods*. Besides these main distinguishing properties, there are also other characteristics that are introduced when they become relevant during the classification of existing AIM attacks in Section 3.5. To get a feel for our classification framework and the herein introduced dimensions, see Section 3.4 for an example.

3.3.1 State of Targeted Transactions

A core goal for permissionless PoW cryptocurrencies is to achieve an (eventually) consistent and totally ordered log of transactions that defines the global state of the shared ledger. Therefore, our classification uses a transaction-centric viewpoint to systematize different attacks and their relation to the underlying consensus. We differentiate between three *states* a transaction can be in from the perspective of a participant (miner or client):

- **unconfirmed**⁵, the transaction has been broadcasted in the respective P2P network;
- **confirmed**, the transaction has been confirmed by at least one block, i.e., has been included in a block;
- **settled**, the transaction has been confirmed by at least k blocks, where k is defined by the recipient of the transaction. We denote $k_{participant}$ to refer to the confirmation policy of a participant if it is not clear from the context, e.g., k_V denotes the confirmation policy of the victim.

3.3.2 Intended Impact/Influence on Transactions

We further separate between the following four main types of how AIM can have an influence on transactions and their ordering:

- **transaction revision**, change a previously proposed, possibly confirmed, or settled transaction;
- **transaction exclusion/censorship**, exclude a specific transaction, or set of transactions, from the log of transactions for a bounded amount of time, i.e., the transaction remains unconfirmed.
- **transaction ordering**, change either the proposed, confirmed, or already settled upon order of transactions;
- **transaction triggering**, incentivize the creation of one or multiple transactions by a specific actor or group of actors, e.g., trigger spending transactions for *anyone-can-spend* outputs.

⁵Sometimes also referred to as *proposed*, or *published* in related literature.

The design paradigms of the underlying cryptocurrency have to be considered to assess the impact and effects of the mentioned manipulation methods. For example, the impact of transaction (re)ordering in a smart contract-capable cryptocurrency is greater than for a cryptocurrency platform that does not support smart contracts. Conversely, the censorship of undesired effect is easier to define programmatically in a UTXO-based model, as there can only be one transaction spending a certain unspent output, compared to a smart contract capable cryptocurrency where a transaction to a smart contract can be routed through several layers of contract invocations. Therefore, influence methods such as transaction ordering and exclusion have a variable impact depending on the targeted platform. Similarly, the ability to invalidate a transaction can result from successfully performing one or more of the above transaction manipulation types. Thereby, the definition of “invalid” depends on the underlying cryptocurrency and is different for UTXO and account-based models. For example, to invalidate a transaction to a smart contract in Ethereum, two approaches exist: Either a transaction is not accepted because a transaction with the same nonce was already included in Ethereum, or the transaction throws an exception during execution because it operates on an (unexpectedly) changed state. The first would be a result of transaction revision, while the latter can happen because of a change in the order of invoked transactions or the exclusion of a previous transaction.

Some AIM attacks may allow multiple types of transaction manipulation at the same time, while others are specifically constructed to support only one method (see Table 3.1). Depending on the state of the targeted transaction(s) (proposed, confirmed, settled), the attack might vary in cost and the required level of interference with consensus.

3.3.3 Required Interference with Consensus

While the previous classification of transaction manipulation attacks describes the intended impact, here we consider the *required interference* with consensus by which they can be achieved. Specifically, we introduce three different fork requirements:

- **Deep-fork required**, where a fork with a depth of at least ℓ exceeding a security parameter k_V is necessary (i.e., $\ell > k_V$). The victim defines k_V [GKL15, SZ16], referring to its required number of confirmation blocks for accepting transactions⁶. In other words, the victim indirectly defines the required minimum fork length ℓ by his choice of k_V .
- **Near-fork required**, where the required fork depth is *not* dependent on k_V , but forks might be required. In other words, the attacker defines the gap k_{gap} (which can be smaller than k_V) he wants to overcome.⁷

⁶We emphasize that each transaction has a recipient (and thus a potential victim with an individual k_V), in practice, there is no global security parameter k that holds for all transactions.

⁷The length of k_{gap} also depends on the attacker’s resources and willingness to succeed (e.g., to exclude a certain block).

- **No-fork required**, where no blockchain reorganization is necessary at all (i.e., $\ell = 0$).

The required interference with consensus specifies the chain reorganization needed. A classical double-spending attack scenario [Ros14, SZ16] can be considered as a *transaction revision* attempt in which a single attacker aims at producing a longer chain (possibly in secret [ES14, SSZ16]) than the main chain to revert one (or possibly many) of *his own* transactions. Therefore, this attack requires *deep-forks* ($\ell > k$) to reorganize the chain. In most models, the attacker is assumed to have full control over the required hashrate to perform the attack, he can also arbitrarily order and exclude transactions from the longest chain. Clearly, an attacker with more than 50% of the hashrate is able to eventually produce a longer chain with probability one and thus can revert/undo any transaction and permanently perform all four kinds of transaction manipulation attacks by providing a longer chain⁸.

No-fork attacks distinguish themselves from the other two categories by aiming to manipulate miner's *block proposals* rather than (preliminary) consensus *decisions*, i.e., already mined blocks. In the context of PoW cryptocurrencies, manipulating a miner's block proposal means influencing the input block used for finding and adding a valid PoW. Deep- and near-fork attacks seek to undo state updates to the ledger that are already confirmed by subsequent PoW.

3.3.4 Payment Method Used

AIM attacks either pay for compliant behavior or penalize for non-compliant actions. How this mechanism is set up depends on the attack in question. There are three general methods that differ in which currency is used for the payment.

- **In-band payment**: The payment is performed in the target cryptocurrency. Most early bribing attacks were designed to gain in-band profits, for example, checklock-time bribes [Bon16], whale transactions [LK17], or history revision contracts in Ethereum [MHM18].
- **Out-of-band payment**: The payment is performed in another currency, the so-called funding cryptocurrency. Some AIM attacks which utilize out-of-band funding were designed as Goldfinger attacks, for example, GoldfingerCon [MHM18] and Pitchforks [JSSW18]. Others can be executed as Goldfinger attack or with the goal of gaining in-band profits, for example, creating alternative mining puzzles [TJS16], or the out-of-band variants of P2W attacks [JSZ⁺19]. This highlights that AIM attacks which are intended to destroy a cryptocurrency, i.e., perform a Goldfinger attack, inherently require methods of out-of-band funding.
- **Threat**: No direct payment is performed, but a credible threat is constructed that non-compliant behavior could lead to losses [sAM, MJP⁺20].

⁸Actually, the heaviest chain by PoW, e.g., in Bitcoin measured in difficulty periods.

3.4 Example Use of Our Classification Framework

Whether an attack is executable with or without a fork depends on the intended impact on transactions as well as on the state of the targeted transaction. For example, transaction revision where the victim accepts $k_V = 0$ (zero confirmations) may be executable as no-fork attacks. Other attacks, such as performing a *double-spend* where the victim has been carefully chosen k_V [SZ16], may require deep-forks because they need to substantially affect consensus and violate the security assumption that the common prefix of the blockchain remains stable. Transaction exclusion (*censorship*) may require near-forks to exclude the latest blocks which include the respective transaction.

With our classification framework, we can map *front-running* [EMC19, DGK⁺20, JSZ⁺19] as an attack that aims to influence transaction ordering, while targeting unconfirmed transactions (state of targeted transactions). Compared to that, the so-called *time-bandit attack* [DGK⁺20] also aims to influence transaction ordering, but targets confirmed or even settled transactions. Note that strictly speaking a time-bandit attack is not AIM, as it does not incentivize other participants to aid the attack, but instead relies on “classic” methods like performing a rental attack to temporarily hold the majority of the hashrate

3.5 Classification of Existing AIM Approaches

Equipped with our generalized attack model and the classification by state of and intended impact on transactions as well as the resulting required interference with consensus, we now inspect and compare existing AIM attacks within this section. Table 3.1 presents an overview of our systematization of existing proposals. Each row represents a different attack (in chronological order of their release) and columns outline respective properties.

	Tx rev.	Tx ord.	Tx excl.	Tx trig.	Required interference with consensus	Attacker hashrate p_B	Rational hashrate p_R	Distracts hashrate	Requires smart contract	Payment	Trustless for attacker	Trustless for collaborator	Subsidy	Compensates if attack fails
Bribery [Cun]	✓	✗	✗	(✓)	Deep fork	$\approx (0, \frac{1}{2})$	$\approx [\frac{1}{2}, 1)$	✗	✗	in-band	✓	~	✗	✗
Dark side attack [Ler]	✓	(✓)	(✓)	✗	Deep fork	$\approx (0, \frac{1}{2})$	$\approx [0, 1)$	✗	✗	in-band	✗	✓	✗	✗
Feather-forks [sAM]	✗	✗	✓	✗	Near-/No forks	$\approx (0, \frac{1}{2})$	$\approx [\frac{1}{2}, 1)$	✗	✗	threat	-	-	-	-
Checkclocktime bribes [Bon16]	✓	✗	✗	(✓)	Deep fork	✗	$\approx [\frac{1}{2}, 1)$	✗	✗	in-band	✓	~	✗	✗
Negative fee miningpool [Bon16]	✓	(✓)	✓	✗	Near-/No- /Deep forks	✗	$\approx [\frac{1}{2}, 1)$	✗	✗	out-of-band	✗	✗	✗	✓
Script Puzzle double-spend [TJS16]	✓	(✓)	✓	(✓)	Deep fork	$(0, \frac{1}{2})$	$1 - p_B$	✓	✗	in-band	~	✗	✗	~
Script Puzzle 38.2% attack [TJS16]	✗	(✓)	✓	?†	Near-/No forks	$[0.382, \frac{1}{2})$	$1 - p_B$	✓	?†	out-of-band	?†	?†	✗	✓
Whale Transactions [LK17]	✓	✗	✗	✗	Deep fork	$(0, \frac{1}{2})$	$1 - p_B$	✗	✗	in-band	✓	~	✗	✗
Proof-of-Stale blocks [LVTS17, VTL17]	-*	-*	-*	(✓)	-*	✗	-	✓	✓	out-of-band	~	✓	✗	✓
Fomo3D game [fom18]	-	-	-	✓	No fork	✗	$[0, 1)$	✗	✓	in-band	✓	✓	✗	~
CensorshipCon [MHM18]	✗	(✓)	✓	(✓)	Near-/No forks	$[\frac{1}{3}, \frac{1}{2})$	$[\frac{1}{3}, \frac{2}{3})$	✓	✓	in-band	~	✗	✓	✗
HistoryRevisionCon [MHM18]	✓	✗	✗	(✓)	Deep fork	✗	$\approx [\frac{1}{2}, 1)$	✗	✓	in-band	✓	~	✓	✗
GoldfingerCon [MHM18]	-	-	✓all	(✓)	No fork	✗	$\approx [\frac{1}{2}, 1)$	✗	✓	out-of-band	✓	✓	✗	✓
Race to the door [Bon18]	-	-	-	✓	No fork	✗	$[0, 1)$	✗	✓	o.o.-band/threat	✓	✓	✗	~
Pitchforks [JSSW18]	-	-	✓all	✗	No fork	✗	$(\frac{1}{3}, 1)$	~	✗	out-of-band	✓	✓	✓	✗
Front-running [EMC19, DGK ⁺ 20]	✗	✓	✗	(✓)	No fork	✗	$(0, 1)$	✗	✗	in-band	✗	✓	✗	✓
Pay per Miner Censorship [WHF19]	✗	✗	✓	-	No fork	✗	1	✗	✓	in-band	~	~	✗	✗
Pay per Block Censorship [WHF19]	✗	✗	✓	-	No fork	✗	1	✗	✓	in-band	~	~	✗	✓
Pay per Commit Censorship [WHF19]	✗	✗	✓	-	Near-/No fork	✗	1	✗	✓	in-band	~	~	✗	✗
P2W Tx Excl. & Ord [JSZ ⁺ 19]	✗	✓	✓	(✓)	Near-/No fork	✗	$[\frac{1}{2}, 1)$	✗	✓	out-of-band	✓	✓	✗	✓
P2W Tx Rev. & Excl. & Ord. [JSZ ⁺ 19]	✓	✓	✓	(✓)	Deep fork	✗	$[\frac{1}{2}, 1)$	✗	✓	out-of-band	✓	✓	✗	✓
P2W Tx Ord. (in-band) [JSZ ⁺ 19]	✗	✓	✗	(✓)	No fork	✗	$(0, 1)$	✗	✓	in-band	✓	✓	✗	✗
P2W Tx Excl. (in-band) [JSZ ⁺ 19]	✗	✗	✓	(✓)	Near-/No fork	✗	$[\frac{1}{2}, 1)$	✗	✓	in-band	✓	✓	✗	✗
BDos [MJP ⁺ 20]	-	-	✓all	✗	Near-/No fork	$\approx [0.21, \frac{1}{2})$ (for BTC)	1	✗/✓	✗	threat	-	-	-	-
HTLC bribing [KNW20, TYE21]	✗	✗	✓	✗	Near-/No fork	✗	1	✗	✗	in-band	✓	~	✗	✗

 Table 3.1: Comparison of existing AIM approaches on cryptocurrencies in *chronological order* according to their appearance.

A property is marked with ✓ if it is achieved and with ✗ otherwise, - is used if a property does not apply. If the symbol is within brackets, e.g., (✓), this means that this property is achieved (or can be augmented), but this was initially not discussed or considered by the authors.

~ means that the property cannot be clearly mapped to any of the previously defined categories without further details or discussion which is given in the textual description.

* means that this attack aims against mining pools and hence is not intended to manipulate the content of the blockchain.

† means that the paper does not explicitly specify the out-of-band payment method but assumes its correctness.

3.5.1 Impact on Transactions

The different ways in which AIM attacks can have an impact on transactions are outlined in Section 3.3.2.

Tx revision: In the first bribing attack, proposed by Bonneau [Bon16], the use of *lock time transactions* is suggested, which are only valid on the attacker’s chain, but there they can be claimed by anyone (anyone-can-spend outputs). Miners are hence expected to be incentivized to mine blocks on the attacker’s chain to collect these bribes as inputs in new transactions included in their new blocks. As a by-product, one transaction per new block is triggered to claim the anyone-can-spend output. Therefore, transaction triggering is technically achieved but set into parenthesis as it is not the main intent of the attack. A variation of the checklocktime bribes which does not trigger additional transactions was proposed by Liao and Katz [LK17] and uses high fee transactions (*whale transactions*) to provide incentives for miners to join the attack. In [MHM18] they proposed a smart contract (`HistoryRevisionCon`) which pays additional in-band rewards to miners of the attacker’s desired Ethereum chain branch, iff the effects of the double-spending transaction have occurred on this branch. Strictly speaking, this attack also triggers transactions as the promised rewards have to be claimed by the bribees from the smart contract. The mentioned attacks ([Bon16, LK17, MHM18]) rely on in-band payments and are designed to replace or *revise* a specific transaction, i.e., perform a single double-spend. As a consequence, they do not consider defining the order or exclusion of arbitrary transactions. Except for the double-spending transaction itself, the block content of subsequent blocks can freely be defined by the bribed miners. Thus – if not explicitly considered – also the blocks produced by the bribed miners will not be fully under the control of the adversary. Therefore, it would be possible for such miners to also perform a double-spend of one of their transactions for free, by piggybacking on the attack financed by the original attacker.

Tx exclusion: There is one notable exception that was specifically designed to exclude transactions: `CensorshipCon` [MHM18] rewards mining uncle blocks to distract the hashrate of bribable miners, which in turn enables the attacker to overtake the Ethereum blockchain s.t., blocks exclusively come from the attacker. Since this attack is in-band, it only works in Ethereum and relies on the uncle block reward scheme of Ethereum to subsidise the attack, i.e., reduce the value of the required bribes. To succeed, it requires that the attacker’s hashrate is larger than $\frac{1}{3}$ and the hashrate of the bribable miners to be between $[\frac{1}{3}, \frac{2}{3})$. If the attack is successful it allows for arbitrary transaction ordering as well and thus also for arbitrary transaction exclusion, as all blocks appended to the main chain during the attack come from the attacker.

`GoldfingerCon` [MHM18] can be seen as a special case of the transaction exclusion attack which rewards Bitcoin miners for mining empty blocks with the help of an Ethereum smart contract. In this case, all transactions are excluded to reduce the utility of the respective cryptocurrency for all its users. Attacks aimed at disrupting entire cryptocurrencies have been described as early as 2012 [BBH⁺12], where the authors highlighted

that collective action of could “Occupy Bitcoin”. Later such attacks to disrupt cryptocurrencies have been termed *Goldfinger attacks* by Kroll et al. [KDF13]. GoldfingerCon was just the first practical instantiation of such an attack. The name is derived from the James Bond movie villain Goldfinger, who seeks to destroy the gold reserves stored in Fort Knox to increase the value of his own holdings. An important aspect of Goldfinger attacks is that the payments have to be performed out-of-band since, if successful, the value of the targeted cryptocurrency is intended to drop. Similarly, *Pitchforks* [JSSW18] leverage merged mining [JZS⁺17] to subsidize the creation of empty (or specially crafted) blocks in the attacked parent chain [JSSW18]. As with all Goldfinger-style attacks, the attacker is required to achieve utility outside of the cryptocurrency economy he wants to attack [KDF13]. In the case of the Pitchfork attack, the external utility comes from a hard-fork, which creates a new cryptocurrency. In this new cryptocurrency, the merge-mined PoW consists of blocks that attack the forked parent cryptocurrency, e.g., are empty. As the hashrate in this case serves an additional use case/application, it is technically not directed anywhere else i.e., not distracted.

Distracted hashrate is redirected from the valid tip(s) of the attacked blockchain to some other form of puzzle or alternative branch, that does not contribute to state transitions of the targeted cryptocurrency. The *Script puzzle* 38.2% [TJS16] and *CensorshipCon* attack [MHM18] are designed to distract the hashrate of bribable miners to gain an advantage over the remaining honest miners. The former redirects the hashrate from the main chain towards puzzles that promise more rewards than honest mining, the latter rewards uncle block mining in Ethereum. The goal of both attacks is that the attacker gains the majority of the hashrate in the respective main chain, and he can hence arbitrarily exclude, or order transactions. Although the attack does not explicitly aim to allow the specific ordering of certain transactions, this capability is achieved as a by-product. Neither attack is reverting blocks to change history, which is a different scenario and requires further analysis in this context, as reverting blocks would change the incentives of miners who have produced them.

Also in *Script Puzzle double-spend* [TJS16] PoW-like puzzles, offering in-band rewards, are published within the respective cryptocurrency with the intent to distract the hashrate of rational miners. Using the gained advantage to overtake the main chain requires an attacker that has direct control over some hashrate. Again, transaction ordering comes as a by-product and was not an explicit design goal, but theoretically, this is the only existing attack utilizing *in-band* payments, which can achieve the three properties: revision, ordering, and exclusion. Although, upon successful execution, rational miners are deprived of their bribes as the previously hidden attack chain becomes the longest chain and does not pay the promised puzzle rewards. This renders the attack non-repeatable against rational miners.

Tx revision (rev.)/ orderomg (ord.) / exclusion (excl.): There are only two proposed attack methods that achieve these three properties in an out-of-band payment scenario: *negative-fee mining pools* [Bon16] and P2W Tx Rev. & Excl. & Ord. [JSZ⁺19]. A negative-fee mining pool is like a classic mining pool, except that it pays out an

above-market return. “*Because such a pool would lose money on expectation, no honest pool should be able to match this reward*” [Bon16]. As with most classic mining pools,⁹ the pool operator can define the content of a block proposal and hence forge arbitrary attack blocks. Even if miners are rational and hence willing to actively participate in such operations, this approach has at least two major limitations: First, miners would still have to trust the pool owner to pay out the promised rewards. Second, miners could report only solutions which are below the current difficulty target (shares) to prove that they are working for the pool but withhold blocks that match the difficulty target. Thereby, they would potentially gain profits by pretending to participate in the attack/pool without actually doing so. This *miner’s dilemma* is a general problem for mining pools [Eya15].

The smart contract design presented in [JSZ⁺19] resolves the limitations of negative-fee mining pools by automating the payment of bribes to complacent miners without requiring any further interaction from the attacker. Thereby, the attacker publishes block templates to the smart contract and offers a bribe for the first miner who can provide a valid PoW solution for such a template. As only payments for valid PoW solutions are provided by the smart contract, it is ensured that the actions of bribees are specifically targeted to aid the attacker. If the attacker deems that the ongoing attack is not likely to succeed, he can stop the investment of further funds by not publishing any further block templates.

Tx triggering: There are only two existing AIM techniques, which are intended to trigger transactions: The *Fomo3D game* [fom18] and the *race to the door* Goldfinger attack sketched by Bonneau [Bon18]. In a race to the door, the attacker “*credibly commits*” to buy out half of all funds present in the targeted cryptocurrency, to utilize them for destroying the system. Therefore, the price the attacker has to pay for those funds is likely to drop the more users decide to sell, increasing the likelihood of the attack to succeed. This creates a vicious cycle, resulting in a race to the door. The idea was not presented in great detail and mainly discussed in the context of overtaking PoS/PoW cryptocurrencies, but of course, such an attack would also trigger sell transactions. Moreover, there are plenty of ways to attack the value of a cryptocurrency while holding substantial amounts of it that are left unexplored.

There are multiple variants of Fomo3D, but roughly the rules are as follows. In this game, which is open for everybody, the last account which has purchased a ticket wins when a timer goes to zero and every purchase again increases the timer by 30 seconds. This leads to the situation that transactions are triggered by rational players as soon as the timer gets close to zero. It was conjectured that the game would never end, but in August 2018 the first round of the game ended and the winner collected 10,469 Ether (\approx \$2.1 M USD at that time)¹⁰. It can be argued that a single instance of this game does not qualify as an “attack”, but the same concept of presumably “free money” available to grab from a smart contract can also be used as an attack method (see our discussion

⁹In P2Pool for example, there is no single operator which can define the content of a block proposal.

¹⁰The winner flooded the network with unrelated high gas transactions to custom smart contracts which congested the network blocking other “last” payments to the game.

in Section 4.6). The interesting aspect about these tx triggering attacks is that they have effects for any hashrate of rational miners as long as there are rational clients. Even if $p_{\mathcal{R}} = 0$, rational clients in the network will issue transactions with the potential to clog new blocks for low fee transactions.

Tx ordering: Dedicated ordering attacks, like front-running [EMC19, DGK⁺20], P2W Tx Excl. & Ord., or P2W Tx Ord. (in-band) [JSZ⁺19], target unconfirmed or confirmed transactions and therefore are cheaper as their interference with consensus is less severe.

3.5.2 Required Interference with Consensus

The concept of required interference with consensus is outlined in Section 3.3.3 and classifies if an attack can be realized with a near-, a deep- or no-fork at all. Depending on the scenario and the desired attack outcome, e.g., if only ordering is relevant, deep-forks are not necessarily required. For example, if the victim accepts unconfirmed transactions, transaction revision can happen without any fork by simply updating the transaction. Bitcoin [Wik] as well as Ethereum allows something like *replace-by-fee* i.e., if there is a transaction signed by the same sender with the same nonce but a significantly higher gas value [Ope], the transaction with the higher gas value replaces the original one in certain clients. This circumstance is also used in the context of *front-running* [EMC19, DGK⁺20]. But front-running is only a subset of possible (re-)ordering attacks, as it might be desirable to place a transaction more accurately in between two other transactions, e.g., as required for exploiting the BlockKing contract [SKH18].

Prior to 2018, ordering attacks on smart contract cryptocurrencies have not been intensively studied [SKH18, KGDS18]. This has recently changed as order fairness has been exposed as a fundamental issue in leader-based consensus protocols [DGK⁺20, KZGJ20, Kur20]. In the context of Nakamoto consensus, every miner that is capable of producing blocks can define the order of the transactions in his blocks. This circumstance alone can be used to gain an advantage in certain scenarios e.g., where transactions race against each other to collect something that is claimable by everybody like an *anyone-can-spend* transaction, the reward of a puzzle, or arbitrage¹¹. But when rational actors are assumed, there are also scenarios where the ordering of transactions can be manipulated by attackers who are not necessarily miners themselves, but have funds at their disposal to launch AIM attacks. In classical front-running, miners are incentivized to prioritize transactions because they carry a larger fee. This however is not a consensus rule and thus lacks enforcement, as even the transaction with the highest fee can still be included at the end of a block, resulting in an all-pay auction if all transactions are eventually included [DGK⁺20]. Although, the overall amount of fee (gas) paid by non-winning transactions can be reduced if the transactions in question can select a cheaper code path if a checkable necessary pre-condition, ensuring the transaction has been included at the desired place, is not met.

¹¹Interestingly the problem of racing transactions was known very early on in the cryptocurrency community, which led to the first fork of Bitcoin, i.e., Namecoin [Nama, JZS⁺17], which introduced a commit reveal scheme to prevent races while registering domain names on the blockchain.

Nevertheless, the attacks observed in practice provide no guarantees for the attacker that the desired ordering is achieved even if the highest transaction fee has been paid [DGK⁺20]. In [JSZ⁺19] an in-band, as well as an out-of-band AIM attack, is proposed, which allows arbitrary transaction ordering while only paying if the desired ordering is observed. Both attacks can be executed without any hashrate assuming rational miners.

3.5.3 Required Hashrate

Required attacker hashrate p_B specifies how much hashrate is required to be under the direct control of the attack (without considering the effects of AIM) for the attack to be successful. As observable in Table 3.1 there are three attacks that require $p_B > 0$. The Script Puzzle 38.2% attack is designed to overtake the blockchain entirely by offering alternative script puzzles with higher rewards to distract the hashrate of rational miners. This allows an adversary with an appropriate hashrate to establish a computational majority and gain a net profit without considering double-spending attacks. In Script Puzzle double-spend the adversary has no explicit minimum hashrate requirement, however low hashrate has to be compensated with more puzzle funds. Moreover, it is designed as a single-shot double-spending attack that, if successful, deprives rational miners of their bribes. Therefore, this particular attack design could only work once with rational miners that fall for it. *CensorshipCon* uses a smart contract to offer in-band bribes for mining uncle blocks to distract hashrate. Thus, it requires some attacker hashrate to include uncle blocks from rational miners in the main chain. Since it has to include all mined uncle blocks, it requires the hashrate of the attacker to be larger than $\frac{1}{3}$ and the hashrate of the bribable miners to be between $[\frac{1}{3}, \frac{2}{3}]$.

It makes sense to bound the attacker hashrate below $\frac{1}{2}$ since otherwise, the attacker has no need to perform bribing attacks as he could overtake the chain single-handedly.

Required rational hashrate p_R specifies how much hashrate is required to be under the control of rational miners for the attack to have a chance to succeed as described and evaluated in the respective paper. Generally, all bribing attacks have to assume that at least some of the miners are rational and hence bribable. Generally, it makes sense to assume that more than half of the miners are rational s.t. attacks have a realistic chance to win longer block races. Both Script Puzzle attacks require all miners to be rational, i.e., $p_B + p_R = 1$, as well as the *Pay per . . .* attacks ($p_R = 1$) described by Winzer et al. [WHF19].

3.5.4 Payment Method

This specifies where the payments to the bribees are performed (see 3.3.4). It can be argued that miners will try not to harm the value of their own cryptocurrency holdings by accepting in-band bribes, hence out-of-band AIM methods are of particular interest. **Subsidy** means that the attack leverages some characteristic of the cryptocurrency, or the environment to become cheaper. In the case of *CensorshipCon*, the rewards from

uncle blocks are used to subsidize the attack, whereas in Pitchforks the additional income from merged mining is used as an incentive.

Compensates if the attack fails refers to the property that at least a portion of the bribe is paid irrespective of the outcome. To successfully engage rational miners, attacks such as Checklocktime bribes [Bon16], Whale Transactions [LK17], and HistoryRevisionCon [MHM18], must pay high rewards in case of success to compensate for the financial risk faced by bribees if the attack fails despite their participation. So far the only attack which facilitates transaction revision that achieves this property is [JSZ⁺19]. Script Puzzle double-spend defrauds the bribed miners if successful and hence actually only pays out rewards if it fails. In front-running attacks, high transaction fees are usually incurred even if the desired ordering effect is not achieved. Thus, in this case, it is also an undesirable property for the attacker. The same holds true for negative-fee mining pools as rewards have to be paid for performed work even if no attack block fulfilling the difficulty target has been submitted by a miner.

3.5.5 Trustlessness

Trustless for the attacker specifies if the attack itself can be exploited by allowing collaborating/bribed miners to profit without adhering to the attack. For example, Script Puzzle attacks require some form of freshness guarantee to prevent bribees from intentionally waiting until the attack fails before computing puzzle solutions to obtain rewards. It is also possible to claim rewards for stale honest blocks that are later on submitted as uncles to the CensorshipCon. Also in naïve front-running attacks the attacker has no guarantee that the desired ordering will be achieved by paying a high fee. The Pay per ... attacks are only modeled theoretically without providing concrete instantiation, therefore they cannot be evaluated in this regard.

Trustless for collaborator specifies if bribees have to trust the attacker that they will receive their payments if they adhere to the attack. In Checklocktime bribes a lock time on individual transaction outputs intends to ensure that they cannot be spent before a particular block height, even by the creator. This ensures that at each height a locked output is released and split into an anyone-can-spend and another locked output. However, the holder of the associated private key can cheat, by creating a conflicting/racing transaction, which also becomes valid after the intended lock time has passed. This conflicting transaction transfers the whole output back to the owner without an additional anyone-can-spend output. However, this attempt is only possible if the attacker is under the control of some hashrate $p_B > 0$, as a miner would never prefer this transaction before the other. The same holds true for Whale Transactions, or HTLC bribes since the attacker has to provide new high-fee transactions for each block on the attack chain at each step of the attack. While HistoryRevisionCon does not explicitly consider trustlessness for collaborating miners, an augmentation is possible¹²,

¹²The issue stems from the fact that the bribing contract checks the balance of the Ethereum account which should receive the bribing funds before issuing any bribes, but without any additional locking constraints, these funds can be moved by the attacker once received.

CensorshipCon requires that the attacker includes blocks produced by collaborating miners as uncle blocks and thus is not trustless. The Script Puzzle double-spend attack is designed as a one-shot attack that defrauds collaborators. The Script Puzzle 38.2% attack does not specify how payments are performed and assumes a working trustless out-of-band payment method.

3.6 Costs, Profits, and Extractable Value

In this section, we want to highlight the challenges of comparing existing AIM attacks with respect to their costs and potential profits.

First of all, the presented attacks differ significantly with respect to their system- and attack models, which have diverse goals regarding their intended influence on transactions (revision, ordering, exclusion, triggering), as well as varying assumptions regarding the capabilities of the attacker, e.g., hashrate and funds.

Second, not all existing proposals have analyzed the involved costs and gains in a comparable way. Attacks such as the Script Puzzle double-spend or CensorshipCon express the required funds in terms of the hashrate which is also required to successfully execute it [TJS16, MHM18]. For transaction revision using Whale Transaction or P2W attacks [LK17, JSZ⁺19], concrete values are provided while at the same time no hashrate is required. In GoldfingerCon [MHM18] only the costs of invoking the smart contract are provided.

Costs: What stands out in the comparison of costs is that: i) Attacks that compensate collaborating rational miners even if the attack fails are cheaper. The reason for this is that such attacks do not have to provide high bribes to account for the risks faced by bribees if the attack is unsuccessful [WHF19, JSZ⁺19]. ii) Attacks that exclusively focus on transaction exclusion or (re)ordering of unconfirmed transactions are substantially cheaper as they only compete with the fee, i.e., extractable value, of the transaction(s) in conflict [WHF19, JSZ⁺19, DGK⁺20, KNW20, TYE21].

Profit: To calculate the profit of the attack it is important to estimate the costs as well as the extractable value. In this context, the term *miner extractable value* [DGK⁺20] has been coined to describe the value which can be extracted by a miner by including a certain transaction in terms of fees, or guaranteed profits through atomic token arbitrage within the context of one transaction. In relation to other AIM attacks surveyed in this chapter, this leads to an interesting observation: We argue that the extractable value of a transaction for a certain party cannot readily be determined by exclusively looking at the cryptocurrency system in which this transaction is to be performed. The reason is that there might be additional protocols like colored coins [Ros12] or out-of-band payments from AIM attacks at play, which can influence the (miner) extractable value of a given transaction. This is an instantiation of a more general observation that game-theoretic analysis is not composable.

The question of whether AIM is profitable can be summarized by comparing the extractable value as well as the costs of the attack and the behavior intended by the protocol designer. The following simplified equation was adapted from Böhme [Böh19].

$$EV(\text{attack}) - \text{costs}_{\text{attack}} > EV(\text{follow protocol}) - \text{costs}_{\text{follow protocol}}$$

Let's assume two unconfirmed, but conflicting Bitcoin transactions (tx_1, tx_2) are competing for a place in the next block. If the extractable fee of one transaction is greater than for the other $\text{Fee}(tx_1) > \text{Fee}(tx_2)$, it would be rational for the miner to include tx_1 , since $EV(tx_1) = \text{Fee}(tx_1)$. But if there is a side payment, due to an AIM attack on a different funding cryptocurrency (e.g., Ethereum) for including tx_2 , which leads to the situation that $EV(tx_2) > EV(tx_1)$, then the situation for the rational miner changes. In this case, the reason for the change is not directly visible in Bitcoin.

Using value of a transaction as an indicator for NC PoW blockchain security is not particularly new and was common practise already in various early analysis approaches, such as the MDP in [GKW⁺16]. In this rigorous analysis, the authors state their rationale behind choosing this parameter as follows:

“We argue that v_d emerges as a robust metric to quantify security under double-spending attacks. Namely, if the reward of honest mining is larger than that of dishonest behaviour, merchants can safely accept a payment transaction of value v_d (since such a value is considered secure, e.g., based on a given confirmation number).” [GKW⁺16]

This statement highlights two implicit assumptions, which have been common in various early analysis approaches of NC security:

- There is only one attacker trying to double-spend a single transaction.
- The value of the transaction in question is solely determined by the value it carries in native currency units.

These assumptions have been challenged by subsequent works. The question of whether it is possible to upper-bound the extractable value was also touched by Budish [Bud18] in a different setting and from the perspective of double-spending attacks only. Under a simplified model, the extractable value of a double-spend is the transferred value of coins¹³. To calculate the required rewards and fees for making double-spending attacks economically unattractive, the author assumed that in the worst case every transaction in a block is potentially up for double-spending and highlights that the relationship between reward and fees, compared to the value transferred in Bitcoin makes such attacks economically feasible in theory. An instantiation of an attack in which every transaction of a block can theoretically become a target for double-spending, has been proposed in [JSZ⁺19], where a crowd-funded attack is described, utilizing smart contracts. The

¹³The dependency between transaction value and confirmation time k_V , is also discussed in [SZ16].

			duration of control	
			temporary	permanent
Source	PoW	new	rent	build
			AIM	AIM
	PoW & PoS	existing	bribe	buy out
			AIM	AIM

Table 3.2: Strategies to gain capacity in Nakamoto consensus according to [Bon18], augmented with AIM strategies (colored background).

goal is to distribute the costs of multiple double-spend attempts in the same block to the set of transacting entities.

From these examples, we see that it is hard or even impossible to accurately bound the extractable value of transactions (and thus blocks) in a multi-cryptocurrency ecosystem by solely looking at data from one cryptocurrency. A related meta argument was presented in [FB19].

3.7 Relation of AIM to other Attack Methods and Areas

We finally discuss the relation of AIM to other methods of gaining capacity in Nakamoto consensus, as well as highlight open questions and directions for future work in this area. By gaining capacity, we refer to the accumulation of voting rights in the permissionless protocol.

Relationship of AIM to other ways of gaining capacity: In the paper [Bon18] an excellent classification of different methods on how to gain capacity in Nakamoto consensus is provided. These methods are separated into: *rent*, *build*, *bribe*, and *buy out*. Hereby, *rent*, *buy out* as well as *build* refers to classical methods of renting hardware, buying cryptocurrency units at exchanges, or building new data centers for mining. We augment this classification and argue that AIM can be used to construct algorithmic ways for all these methods. Table 3.2 depicts an augmented version from [Bon18] showing the different methods of how to obtain capacity in Nakamoto consensus.

According to the original classification in [Bon18], bribing is a *temporary* attack, which utilizes the *existing* resources of miners. If the terms *new* and *existing*, in the context of PoW capacity, are to be interpreted from the perspective of the targeted system, then some existing attacks which rely on out-of-band payments would also classify as *rent*. The reason for this is: They are also able to attract new capacity currently bound in other cryptocurrencies which utilize the same PoW algorithm, like [TJS16, JSSW18, JSZ⁺19]. Capacity which is present in a different cryptocurrency is also new to the targeted cryptocurrency if miners decide to switch to supporting an attack.

We further argue that *buy out* attacks can theoretically be done algorithmically using cross-chain atomic swaps [Her18] (or any other blockchain interlinking protocol). A

race-to-the-door style attack [Bon18] in combination with cross-chain atomic swaps can be imagined to perform Goldfinger-style attacks on smart contract capable PoW cryptocurrencies. Hereby, out-of-band payments are used to buy out cryptocurrency units, through a smart contract, which is going to use these previously bought cryptocurrency assets to perform a denial of service attack by dumping the previously bought crypto assets on the market as freely available for anyone to claim after a certain timeout. If there is a limit for what is claimable per transaction, as well as the requirement of a high fee, this on-chain faucet construction will trigger a flood of transactions as soon as the timeout is reached. In this case, existing funds are bought and permanently redistributed with the intent to perform a denial-of-service attack and at the same time collapse the market due to increased supply.

It remains to be shown that it is theoretically possible to build *permanent* AIM attacks. Arguably, any Goldfinger attack, such as GoldfingerCon [MHM18], which creates enough external utility to refuel the attack, can in theory be constructed in a way to run permanently. Although, it is unlikely that a Goldfinger attack has to be continued infinitely long if the intended effects have already occurred. An attack that also discusses its perpetuity is the Script puzzle 38.2% attack. In this case, the attack can also theoretically be used to permanently overtake the chain by supplying puzzles that provide out-of-band rewards and thereby overtake the original blockchain with 38.2% of the total hashrate. Also, the pitchfork [JSSW18], in which the additional revenue stream to sustain the attack comes from a fork of the targeted cryptocurrency and not from a previously determined bribing fund, can in theory be sustained infinitely long. Whether the attack can be sustained depends on the value of the newly generated cryptocurrency. An interesting analogy exists between any permanent AIM attack and a cryptocurrency itself. From the perspective of a miner who exclusively mines on puzzles for any of these three permanent attacks, there is no difference to mining on any other PoW-based cryptocurrency other than the format of the associated PoW.

Mitigation and counter-attacks: The presented systematization has a very attack-centric view on the issue at hand. This is due to the selection of papers, which almost all have a very attack-focused viewpoint. Therefore, countermeasures and counter-attacks are often omitted in these papers, or not discussed to a great extent.

Nevertheless, for the victim(s) counter-bribing might be a viable strategy against AIM. The difficulty of successfully executing counter-bribing highly depends on the respective scenario. In the end, counter-bribing can also be countered by counter-counter-bribing and so forth. Therefore, as soon as this route is taken, the result becomes a bidding game. Against transaction exclusion attacks, counter-bribing can be performed by increasing the fee of the transaction to be excluded such that it surpasses the value promised for not including the transaction. If defenders have imperfect information, they may not be able to immediately respond with counter-bribes. In this case, some of the attack chain blocks may have already been mined, or even taken the lead, before they are recognized by defenders. Counter-bribing then necessitates a fork, and thus a more expensive transaction revision attack, leading to asymmetric costs in the bidding game.

This illustrates an important aspect of AIM, namely its visibility. On the one hand, sufficiently many rational miners of the targeted cryptocurrency have to recognize that an attack is occurring, otherwise they won't join in and the attack is likely to fail. On the other hand, if the victims of the attack recognize its existence, they can initiate and coordinate a counter-bribing attack. So the optimal conditions for AIM arise if all rational miners have been informed directly about the attack, while all victims/merchants do not monitor the chain to check if an attack is going on and are not miners themselves. If the payments are made out-of-band, they are rendered more stealthy to victims who only monitor the targeted cryptocurrency. It can hence be argued that counter-attacks by victims are harder to execute as they are not immediately aware of the bribing value that is being bet against them on a different funding cryptocurrency. We also follow the argument in [Bon16] that requiring clients to monitor the chain and actively engage in counter-bribing is undesirable, and out-of-band attacks further amplify this problem as clients would have to concurrently monitor a variety of cryptocurrencies.

To prevent repercussions, participating miners can make use of the fact that the PoW mining process itself does not require any strong identity by using different payout addresses. Of course, their received rewards can be traced, but available privacy techniques could be used to camouflage the real recipient of the funds, e.g., [RMSK14, MM17, HBG16].

In particular with regard to AIM attacks the question remains if more powerful attacks, in terms of their capabilities as well as their incentive structures, are possible. We answer this question in the affirmative in Chapter 4, where we propose new versatile AIM attacks.

Pay To Win Attacks

*“The system is secure as long as **honest** nodes collectively control more CPU power than any cooperating group of attacker nodes.”*

– Satoshi Nakamoto [Nak08]

(emphasis added)

In this chapter¹, we extend the attack landscape of bribing attacks on cryptocurrencies by presenting a new method, which we call *Pay-To-Win* (P2W). To the best of our knowledge, it is the first approach capable of facilitating double-spend collusion across different blockchains. Moreover, our technique can also be used to specifically incentivize transaction exclusion or (re)ordering. For our construction, we rely on smart contracts to render the payment and receipt of bribes trustless for the briber as well as the bribee. The attacks described in this chapter are operated and financed *out-of-band* i.e., on a funding cryptocurrency, while the consequences are induced in a different target cryptocurrency². Hereby, the main requirement is that smart contracts on the funding cryptocurrency are able to verify the consensus rules of the target. For a concrete instantiation of our P2W method, we choose Bitcoin as a target and Ethereum as a funding cryptocurrency. Our P2W method is designed in a way that reimburses collaborators even in the case of an unsuccessful attack. Interestingly, this actually renders our approach approximately one order of magnitude cheaper than comparable bribing techniques (e.g., the whale attack). We demonstrate the technical feasibility of P2W attacks by publishing all relevant artefacts, ranging from calculations of success probabilities to a fully functional proof-of-concept implementation, consisting of an Ethereum smart contract and a Python client³.

¹This chapter represents an extended version of publication [JSZ⁺19].

²We also provide two incentive manipulation attacks which utilize *in-band* payments in the Appendix A.6 and A.5.

³https://github.com/kernoelpanic/pay2win_artefacts

4.1 Relation of Pay To Win to Other AIM Attacks

Despite an ever-growing body of research in the field of cryptocurrencies, it is an open question if Bitcoin, and thus Nakamoto consensus, is incentive compatible under practical conditions, i.e., that the intended properties of the system emerge from the appropriate utility model [Bon16, Bon18]. *Bribing attacks*, in particular, target incentive compatibility and assume that at least some of the miners act *rationally*, i.e., they accept bribes to maximize their profit. If the attacker, together with all bribable miners, can gain a sizable portion of the computational power, even for a short period of time, attacks are likely to succeed.

Since the first descriptions of bribing attacks [Cun, Bon16], various attack approaches, which tamper with the incentives of protocol participants, have been presented for different scenarios and models. As bribing [TJS16, LK17, MHM18, WHF19, KNW20], front-running [KNS⁺18, EMC19, DGK⁺20] Goldfinger [KDF13, JSSW18, Bon18] and other related attacks, all intend to manipulate the incentives of rational actors in the system, we jointly consider them under the general term *algorithmic incentive manipulation* (AIM). So far, most proposed AIM attack strategies focus on optimizing a player's (miner's) utility by accepting *in-band* bribes, i.e., payments in the respective cryptocurrency [Bon16, LK17, MHM18, WHF19, KNW20]. Thus, a common argument against the practicality of such attacks is that miners have little incentive to participate, as they would put the economic value of their respective cryptocurrency at risk, harming their own income stream. Another common counterargument against in-band bribing attacks is that they are considered expensive for an adversary (e.g., costs of several hundred bitcoins for one successful attack [LK17]), or require substantial amounts of computing power by the attacker.

In this chapter, we present an AIM attack method called *Pay-To-Win* (P2W), which generalizes the construction of different AIM attacks on PoW Cryptocurrencies by leveraging smart contract platforms. Our attack requires no attacker hashrate, and an order of magnitude fewer funds than comparable attacks (i.e., the whale attack). To highlight the technical and economical feasibility of our approach, we provide a concrete instantiation of our P2W design, representing a new bribing attack. It uses a smart contract capable funding cryptocurrency (Ethereum) to finance and operate an attack on a (different) target cryptocurrency (Bitcoin). All bribes are paid in the funding cryptocurrency, i.e., out-of-band. Prior to our attacks, out-of-band payments have only been used in the context of Goldfinger attacks, where the goal of an attacker is to destroy a competing cryptocurrency to gain some undefined external utility [KDF13]. The attacks we present in this chapter can be performed based on either strategy, using in-band profit, or as out-of-band Goldfinger-style attacks to destroy the value of the targeted cryptocurrency. In a multi-cryptocurrency world, P2W attacks demonstrate that utilizing out-of-band payments can pose an even greater threat to cryptocurrencies, as the argument that miners won't harm their own income stream must be critically examined in this context. Consider as an example two PoW cryptocurrencies that share the same PoW algorithm and have competing interests, for example, Bitcoin and Bitcoin

Cash. If rational Bitcoin miners face the opportunity of earning Ether for performing attacks on Bitcoin Cash, they may be willing to redirect their hashrate for this purpose, especially if they are guaranteed to receive the promised out-of-band rewards/bribes.

We show that such sophisticated trustless *out-of-band* attacks on Bitcoin-like protocols can readily be constructed, given any state-of-the-art smart contract platform capable of verifying the consensus rules of the target for the duration of the attack. Moreover, we show that the cost for an attacker can be considerably reduced by guaranteeing that participating bribees are reimbursed, as well as by aligning the interests of multiple attacks (crowdfunding) in a trustless manner, e.g., through smart contract code. Furthermore, cross-chain transaction ordering attacks can also be executed as targeted bribing attacks using our method. This possibility for rational miners to (trustlessly) auction the contents of their block proposals (i.e., votes) to the highest bidder raises fundamental questions on the security and purported guarantees of most permissionless blockchains.

4.1.1 Contribution

We propose a new design pattern, called *Pay-To-Win* (P2W), for out-of-band algorithmic incentive manipulation (AIM) attacks. To highlight the concepts behind our design approach, we provide instantiations for two *new AIM attacks* (Section 4.4 and 4.5).⁴ Both are trustless for the attacker and the collaborating miners, rely on out-of-band payments in a different cryptocurrency and do not require the adversary to control any hashrate. The first instantiation incentivizes deep-forks and double-spend collusion. The second instantiation has fewer capabilities but is cheaper as it does not require deep-forks, or in the best case even no-forks at all. On the technical level, We introduce three crucial enhancements to AIM attacks: (i) *ephemeral mining relays*, as an underlying construction which is required to execute our trustless, time-bounded, cross-chain attacks, (ii) *guaranteed payment* of bribed miners even if the attack fails, which actually reduces the costs of such attacks, and (iii) *crowdfunded attacks*, to further reduce the individual cost of executing such attacks. Our contributions are as follows:

- P2W attack method to guarantee payments to participating bribees
- An instantiation for a trustless out-of-band AIM attack to incentivize double-spend collusion.
- An instantiation for a trustless out-of-band AIM attack to incentivize transaction exclusion and/or ordering
- An approach to crowdfund out-of-band double-spending attacks

⁴We also describe and evaluate two new in-band attacks targeting transaction ordering and transaction exclusion in the Appendix A.6 and A.5. The latter (in-band transaction exclusion) was also described and analyzed in concurrent work by Winzer et al. [WHF19], but no concrete instantiation was given.

- Concrete cost estimates for all our attacks, as well as a PoC of our attack smart contract to demonstrate the feasibility and estimate operational costs. All artefacts reaching from calculations and simulations to the PoC are available online ⁵.

4.2 Model

We focus on *permissionless* [Vuk15] proof-of-work (PoW) cryptocurrencies, as the majority of related bribing attacks target Bitcoin, Ethereum, and systems with a similar design. That is, we assume protocols adhering to the design principles of Bitcoin, or its backbone protocol [Nak08, GKL16, PSS17], which is sometimes referred to as Nakamoto consensus [DKT⁺20, SJS⁺18]. Within the attacked cryptocurrency we differentiate between *miners*, who participate in the consensus protocol and attempt to solve PoW-puzzles, and *clients*, who do not engage in such activities. Following the models of related work [LK17, TJS16, MHM18, Bon16], we assume the set of miners to be fixed, and their respective computational power within the network to remain constant.

To abstract from currency details, we use the term *value* as a universal denomination for the purchasing power of cryptocurrency units, or any other out-of-band funds such as fiat currency. Miners and clients may own cryptocurrency units and are able to transfer them (i.e., their value) by creating and broadcasting valid transactions within the network. Moreover, as in prior work [TE18, LK17, MHM18], we likewise make the simplifying assumption that exchange rates are constant over the duration of the attack.

We split participating miners into three groups and their roles remain static for the attack duration. Categories follow the *BAR* (*Byzantine, Altruistic, Rational*) [AAC⁺05, LCW⁺06] [LCW⁺06] behavior model. Additionally, we define the *victim(s)* as another group or individual without computational power, i.e, hashrate.

- **Byzantine miners or attacker(s) (Blofeld)**: The attacker B wants to execute an incentive attack on a *target cryptocurrency*. B is in control of bribing funds $f_B > 0$ that can be in-band or out-of-band, depending on the attack scenario. He has some or no hashrate $p_B \geq 0$ in the target cryptocurrency. B may deviate arbitrarily from the protocol rules.
- **Altruistic or honest miner(s) (Alice)**: Altruistic miners A are honest and always follow the protocol rules, hence they will not accept bribes to mine on a different chain-state or deviate from the rules, even if it would offer a larger profit. Such control some or no hashrate $p_A \geq 0$ in the target cryptocurrency.
- **Rational or bribable miner(s) (Rachel)**: Rational miners R controlling hashrate $p_R > 0$ in the target cryptocurrency They aim to maximize their short-term profits in terms of *value*. We consider such miners “bribable”, i.e., they follow strategies that deviate from the protocol rules as long as they are expected to yield higher

⁵https://github.com/kernoelpanic/pay2win_artefacts

profits than being honest. For our analyzes, we assume rational miners do not concurrently engage in other rational strategies such as selfish mining [ES14].

- **Victim(s) (Vincent):** The set of victims or a single victim, that loses value if the bribing attack is to be successful. The victims control zero hashrate and therefore can be viewed as a client.

It holds that $p_B + p_A + p_R = 1$. The assumption that the victim of an AIM attack has no hashrate is plausible, as the majority of transactions in Bitcoin or Ethereum are made by clients which do not have any hashrate in the system they are using.

Whenever we refer to an attack as *trustless*, we imply that no trusted third party is needed between the briber and bribee to ensure correct payments are performed for the desired actions. Thus the goal is to design AIM in a way that the attacker(s), as well as the collaborating miners, have no incentive to betray each other if they are economically rational.

4.2.1 Communication and Timing

Participants communicate through message passing over a peer-to-peer gossip network, which we assume implements a reliable broadcast functionality. As in previous bribing attacks, we further assume that all miners in the target cryptocurrency have *perfect knowledge* about the attack once it has started. Analogous to [GKL16], we model the adversary Blofeld as *rushing*, meaning that he gets to see all other players' messages before he decides his strategy, e.g., executes his attack. While the attack is performed on a *target cryptocurrency*, the distinct *funding cryptocurrency* is used to orchestrate and fund it. We also assume that the difficulty and the mean block interval of the funding chain are fixed for the duration of the attack and that no additional attacks are concurrently being launched against either cryptocurrency.

4.3 P2W Attack Method

In this section, we introduce a new approach for algorithmic incentive manipulation attacks, which we call *Pay-To-Win* (P2W). The basic idea of P2W is it to design a profitable attack in a way that the (economically) most rational action of a potential bribee is exactly the one desired by the attacker. To achieve this, our approach relies on smart contracts and the specification of *block templates* by the attacker. These templates define the desired block structure for which Blofeld is willing to provide rewards in form of bribes. We consider *out-of-band* attacks to be technically more challenging, as well as more powerful regarding their capabilities (see below), therefore we focus on out-of-band attacks in this chapter. Moreover, the description and evaluation of two new in-band attacks can be found in Section A.5 and Section A.6. The latter attack, describing the incentivisation of transaction exclusion using in-band payments, was also described and

analyzed in concurrent work by Winzer et al. [WHF19], but no concrete instantiation was given.

In cases where the payment is performed *out-of-band*, we differentiate between a *target cryptocurrency*, where the attack is to be executed, and a *funding cryptocurrency*, where the attack is coordinated and funded. While the funding cryptocurrency must support sufficiently expressive smart contracts, there are no such requirements for the target cryptocurrency. For presentation purposes, we choose Bitcoin as the target and Ethereum as the funding cryptocurrency to instantiate and describe our attacks. Theoretically, the attack can be funded on *any* smart contract-capable funding cryptocurrency, which fulfils the requirements listed in Section 4.6.2. This advantage of being fund- and operable on any appropriate smart contract-capable cryptocurrency renders these P2W attacks arguably more difficult to detect and protect against, as the victim(s) would have to monitor multiple, if not all, possible funding blockchains. Moreover, our attacks can also use additional privacy-preserving techniques available on the funding cryptocurrency (e.g., [MM17, Hom]) to hinder the traceability of funds and transactions of involved parties, providing another reason why our attacks can be considered more stealthy compared to attacks utilizing in-band payments. Another advantage of out-of-band payments is, that they are not bound to the exchange value of the targeted cryptocurrency and thus can also be used for Goldfinger-style attacks [JSZ⁺21b, KDF13, Bon18], as the assumption that miners of the target cryptocurrency would not harm their own revenue channel does not necessarily hold true anymore. This is an even more compelling argument in a world where multiple cryptocurrencies either share the same PoW algorithm, or hardware can be effectively used for mining other forms of PoW.

We present two instantiations of attacks utilizing our P2W approach: *P2W Tx revision/exclusion/ordering* and *P2W Tx exclusion/ordering*. Both attacks differ regarding their capabilities as well as their costs. The first also allows to revise already confirmed transactions and thereby facilitates double-spend collusion, while the second is only capable of incentivizing the exclusion as well as the ordering of transactions, but therefore is substantially cheaper. What both instantiations have in common, is that their construction requires a combination of a smart contract-based mining pool [VTL17, LVTS17] and a chain relay [But16, ZHL⁺19, Rel]. We call this underlying construction an *ephemeral mining relay* (EMR). The EMR is introduced and evaluated when explaining our first attack. It takes care that the promised rewards are only paid to complacent bribees who have actively contributed to the attack. Therefore, the two introduced attacks can be considered *trustless*, both for the attacker as well as the collaborating bribed miners. Moreover, our attacks do not require the adversary to control any hashrate, i.e., we assume $p_B = 0$.

To demonstrate the feasibility of our approach and the described attack, we implemented a fully functional prototype of our most powerful attack and evaluated its costs in Ethereum. The source code and all other artefacts related to the evaluation can be found on Github ⁶.

⁶, ⁷

4.4 P2W Transaction Revision, Exclusion and Ordering Attack

To illustrate all underlying concepts, we start with a description of our most powerful attack, which allows for transaction revision and thus directly facilitates double-spend collusion. While we focus on transaction revision in our description, the presented attack also bears the possibility of transaction exclusion or ordering.

On a high level, miners are offered bribes in a funding cryptocurrency (in our case Ethereum) to mine blocks on the favoured branch of a target cryptocurrency (in our case Bitcoin) in which the adversary is executing a double-spend. Moreover, we show how the attack can be constructed to always reward collaborating miners, regardless of the outcome of the attack. Interestingly, this renders our approach significantly cheaper than comparable attacks [LK17]. As a modification to reduce the costs, we also describe how smart contracts can be used to crowdfund and/or combine multiple double-spending attempts into a single coordinated attack, which further reduces the costs for participants. This implies that theoretically all of the transactions in a target's block could be double-spending attempts by the different entities which performed them.

To execute our attacks, Blofeld must construct a smart contract which temporarily rewards the creation of attacker-defined blocks on the target cryptocurrency. We call this technique an *ephemeral mining relay* (EMR) and evaluate its construction at the end of this section. An EMR requires some main attacker for initialization, after which it can be used by him as well as by other collaborating miners/attackers/bribees to coordinate the attack and manage the investment and payout of funds.

4.4.1 Description

Figure 4.1 shows the different stages of the attack on the funding cryptocurrency, as well as two different outcomes (Failed and Successful attack) on the target cryptocurrency. The paid-out compensations (block rewards normalized to 1) and bribes (ϵ) are listed above the respective blocks. The different stages are as follows:

Initialization Phase (deploy, init)

First, the attacker (Blofeld) creates the uninitialized attack contract and publishes it on the Ethereum blockchain. This is done with a *deploy* transaction included in some Ethereum block e_0 from an Ethereum account controlled by the attacker⁸. Then, Blofeld creates a conflicting pair of Bitcoin transactions. The spending transaction tx_B is published on the main chain in Bitcoin immediately, and the double-spending transaction

⁸It is also possible to deploy and initialize the attack contract at the same time (e_1), but publishing an uninitialized attack contract upfront ensures that potential collaborators can audit it and familiarize themselves with the procedure. In any case, it is important that the double-spend transaction tx'_B is disclosed after block b_{k_V} on the main chain, as otherwise Vincent may recognize the double-spending attack and refuse to release the goods.

tx'_B is kept secret. After the confirmation period of k_V blocks (defined by the victim) has passed on the Bitcoin main chain, Blofeld releases an initialization transaction, which defines the conditions of the attack in the smart contract on the Ethereum chain. The block e_1 represents the first block on the Ethereum chain after the Bitcoin block b_{k_V} has been published.

In e_1 the contract is initialized with $k_V + 1$ new Bitcoin block templates, each carrying the transactions from the original chain to collect their fees, but instead of tx_B , the conflicting transaction tx'_B is included. Collaborating miners are now free to mine on these block templates. For the first template, they are only allowed to change the nonce and the coinbase field to find a valid PoW and include their payout Ethereum address in the coinbase. This prevents front-running of solutions (see Section 4.4.1). Once a solution has been found, it has to be submitted by the respective miner to the attack contract, which verifies the correctness of the PoW and that only allowed fields (nonce and coinbase) have been changed. After the first block (b'_1) in the sequence, also the previous block hashes of subsequent blocks ($\{b'_2, \dots\}$) have to be adjusted by collaborating miners. If a submitted solution is valid, the contract knows which previous block hash it must use to verify the next solution and so forth.

As soon as the attacker becomes aware that a valid solution was broadcasted in the Ethereum P2P network, he uses the PoW solution to complete the whole block and submits it to the Bitcoin P2P network. Blofeld and the collaborating miners have an incentive to submit solutions timely, as collaborating miners want to collect an additional bribe ϵ in case the attack succeeds, and the attacker wants to get his blocks included in the Bitcoin main chain to receive the Bitcoin block rewards to his Bitcoin address, and in the best case, perform a successful double-spend.

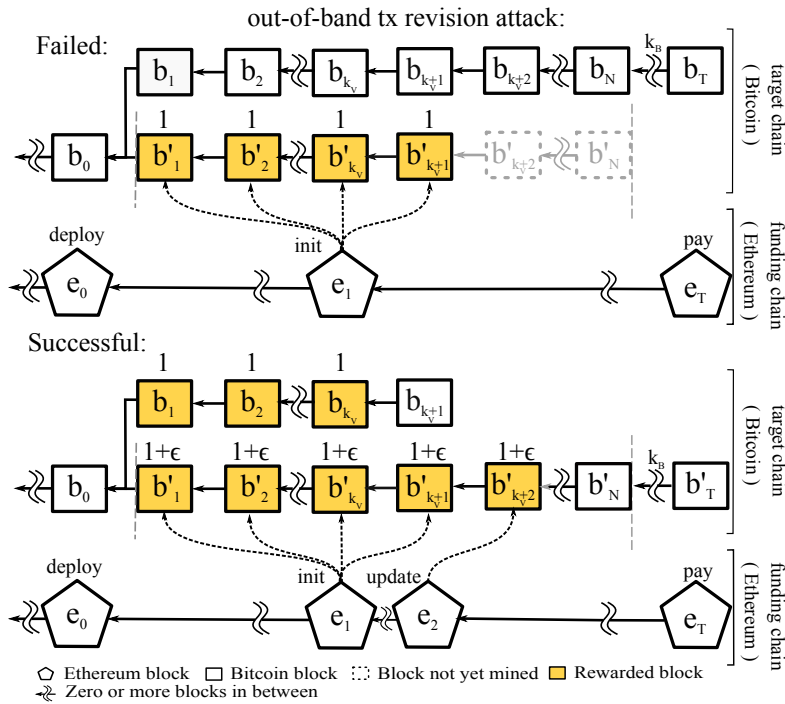


Figure 4.1: Example timeline showing blockchain structure and resulting payouts of a failed, and a successful tx revision attack with out-of-band payments. After k_V blocks on the target chain have passed, the attack contract is initialized with (at least) $k_V + 1$ block templates. The double-spend transaction(s) are included in block b'_1 . The payouts are performed in block e_T . The colored blocks are rewarded by the attack contract, either with their original value (reward + fee normalized to 1) or with an additional ϵ if the attack was successful. The numbers above colored blocks indicated those normalized rewards. If the attack succeeds, the first k_V blocks on the Bitcoin main chain also have to be compensated to provide an incentive for the respective miners (of those blocks) to also mine on the attack chain.

Attack Phase (update)

Bribed miners now proceed to mine $k_V + 1$ blocks on the attack chain. If additional blocks are found on the main chain, the attacker can update the attack contract with new block templates for blocks $k_V + 2$ to N , where N is the maximum number of attack blocks that can be funded by the adversary. Note that N is not necessarily known by Vincent, Rachel or any other observer.

Payout Phase (pay)

The payout phase starts as soon as the attack phase has ended. This happens when k_B blocks have been mined on top of the last block for which a block template has been provided to the smart contract. In the best case, this happens at block $T = k_V + 1 + k_B$, but in our example one update with an additional block template was required, leading

to $T = k_V + 2 + k_B$. The delta of k_B is a security parameter defined by the attacker, which should ensure that every participant had enough time to submit information about the longest Bitcoin chain to the contract and that the sequence of blocks relevant for the attack has received sufficient confirmation blocks⁹.

The attack terminates as soon as the first block of height T is committed to the contract. This can be a block of building on top of the original chain, or the attack chain. After the attack has terminated, the contract unlocks the payment of compensations and rewards for the miners of the associated blocks. Now all miners who joined the attack and contributed blocks can collect their compensations and/or bribes from the contract. To accurately pay out funds, the contract on Ethereum has to determine which chain in Bitcoin has won the race and is now the longest chain. Thereby, the contract has to distinguish between two possible outcomes:

- *Attack failed (original chain wins)*. In this case, the contract must compensate the bribed miners for their contributed blocks to the attack chain, which are now stale. These are at most $\{b'_1, \dots, b'_N\}$, Every collaborating miner who mined and successfully submitted a block to the attack contract receives the reward for that block without an additional ϵ .
- *Attack succeeded (attack chain wins)*. If the attack chain wins, then the contract executes the following actions: 1) Fully compensate the miners of k_V main chain blocks starting from b_1 , which are now stale. This is necessary to provide an incentive also for those miners to switch and contribute to the attack chain, as they otherwise would lose their rewards from blocks they contributed to the original chain if the attack is successful. 2) Pay the miner of every attack chain block, (b'_1 to b'_{k_V+2} in our example max. till b'_N), the full block reward plus an additional ϵ as a bribe in Ether.

Upon being invoked with a miner's cash-out transaction, the contract checks if the attack has already finished, i.e., a valid chain up to block height T is known, and which chain has won the race. Then the contract pays out accordingly.

Incentives to Submit Information

Since collaborating miners are competing for mined attack chain blocks and want the attack to be successful to receive the additional bribes, they have an incentive to submit their attack chain blocks to the attack timely. Additionally, Blofeld who initialized the contract and provided the funds has an incentive to submit the relevant part of the competing original chain, if such a conflicting longer chain ($\{b_1, \dots, b_T\}$) exists, since he would pay an additional ϵ for every block otherwise. Therefore there is always some actor who has an incentive to submit the correct longest chain to the attack contract.

⁹Ideally k_B is specified as an acceptance policy logarithmic in the chain's length as described in [SZ16].

Ethereum Payout Address Derivation

To determine the correct Ethereum payout addresses of collaborating miners, the following approaches are feasible. In the simplest case, all bribed miners directly provide their Ethereum address in the coinbase field of every submitted Bitcoin block on the attack chain. Alternatively, they can disclose their public-keys directly via *pay-to-pubkey* outputs in the coinbase transaction in Bitcoin. The Bitcoin address public-key can then be used to derive the corresponding Ethereum address, as described and implemented in the Goldfinger attack example in [MHM18].

For the first k_V main chain blocks, where miners were not yet aware of the attack, they must prove to the contract that they indeed mined the respective block(s). This can be achieved, e.g., by providing the ECDSA public-keys corresponding to the payouts in the respective coinbase outputs to the smart contract, such that it can check if they match and then recompute the corresponding Ethereum address.

4.4.2 Evaluation with Solely Rational Miners

($p_R = 1$)

As rational miners will participate in the attack as long as it is expected to yield more profit than honest mining, the remaining question is, what budget (r_{budget}) in Ether is required by Blofeld for the attack to succeed. As the Bitcoin block rewards and bribes have to be paid out in Ether, we assume a fixed exchange rate between cryptocurrencies to derive our lower bound in terms of BTC required.

Blofeld has to lock funds in the attack contract for each submitted block template, to ensure complacent miners can be certain to receive their rewards if they submit blocks and thus are incentivized to join the attack. Therefore, the duration of the attack is the main driver of the required budget. As the duration is dependent on the security parameter k_V chosen by Vincent, $N > k_V$ has to hold for an attack to be feasible.

Necessary Attack Budget

For Bitcoin, a common choice is $k_V = 6$ requiring N to be at least 7. The budget of the attack contract must cover all rewards which could potentially be paid out by the contract. For the most expensive case, which is a successful attack, this encompasses: The bribes (ϵ) as well as Bitcoin block rewards including fees¹⁰ (r_{reward}), which we previously normalized to 1 in Figure 4.1. Assuming the current block reward (6.25 BTC), average fees (≈ 2 BTC), operational costs ($cost_{operational} = 0.5$ BTC), as well as a bribe of $\epsilon = 1$ BTC, this leads to a budget of 114.75 BTC which has to be provided to the attack contract in Ether upfront:

$$r_{budget} = k_V \cdot r_{reward} + N \cdot (r_{reward} + \epsilon) + cost_{operational} \quad (4.1)$$

¹⁰In a concrete attack of course r_{reward} is not constant, but given by the coinbase output values of every submitted block.

As Blofeld receives the Bitcoin block rewards in case of a successful attack, the actual costs of the attack are *much smaller* than the required budget Blofeld has to lock in the contract.

Costs and Profitability of a Successful Attack

If the attack is successful, then Blofeld earns the block rewards on the attack chain in BTC which compensates his payouts to bribed miners in Ether. The costs for a successful attack are thus reduced by $N \cdot r_{reward}$ attack chain blocks, whereas rewards must be paid for $N \cdot (r_{reward} + \epsilon)$ block templates. The remaining costs of a successful attack stem from the $k_V \cdot r_{reward}$ original chain blocks that have to be compensated on the attack chain.

$$cost_{success} = k_V \cdot r_{reward} + N \cdot \epsilon + cost_{operational} \quad (4.2)$$

The initial k_V compensations are necessary to provide the same incentive for *all* miners that have already produced blocks on the main chain to switch to the attack chain. Since we assume rational miners, the attack in this scenario is always successful if $N > k_V$ and $\epsilon > 0$ hold. For Bitcoin, this means that the costs of a successful double-spend with $k_V = 6$ and $r_{reward} = 8.25$ and $\epsilon = 1$ are $cost_{success} = 57$ BTC. For a successful attack to be profitable, the value of the double-spend has to be greater than this value. In Bitcoin, transactions carrying more than 57 BTC are observed regularly¹¹. For comparison, in its cheapest configuration, the whale attack costs approximately 770 BTC [LK17], but it was simulated for a previous Bitcoin reward epoch, where block rewards have been higher. Even if we assume $r_V = 12.5$ BTC, our attack would cost 94.5 BTC, which is considerably lower than the whale attack. The remaining difference in our approach is that the whale attack does not assume all miners to be rational. In Section 4.4.3 we also extend our evaluation to this model by introducing altruistic miners.

Costs of a Failed Attack

Although the attack cannot fail in a model where all miners are rational and the attacker has enough budget, it is relevant for a scenario where $p_{\mathcal{R}} < 1$ to determine the worst-case cost for an unsuccessful attack. In the worst case, the attack duration is N and not a single block produced by complacent miners (according to a published block template) made it into the main chain. Then the costs are determined by the duration N and the block rewards including fees (r_{reward}):

$$cost_{fail} = N \cdot r_{reward} + cost_{operational} \quad (4.3)$$

Setting the same values for r_{reward} and N amounts to approximately $cost_{fail} = 58.25$ BTC in our example.

¹¹cf. [https://blockchair.com/bitcoin/outputs?s=value\(desc\),time\(desc\)&q=time\(2020-10\),value\(6000000000..\)#](https://blockchair.com/bitcoin/outputs?s=value(desc),time(desc)&q=time(2020-10),value(6000000000..)#)

4.4.3 Evaluation with Altruistic Miners

$$(p_A > 0 \wedge p_R + p_A = 1)$$

We now discuss a more realistic scenario where not all miners switch to the attack chain immediately, i.e., some of them act altruistically. Altruistic miners follow the protocol rules and only switch to the attack chain if it becomes the longest chain in the network – but do not attempt to optimize their revenue, contrary to economically rational, i.e., bribable, miners¹².

We derive the probability of the attack chain winning a race against altruistic miners, based on the budget of the attacker and the initial gap between those chains which has to be overcome k_{gap} where k_{gap} is initially set to $k_{gap} = k_V$. The difference between k_V and k_{gap} is that k_{gap} can increase when altruistic miners find a new block, while k_V is static. In other words, the attack chain must find $k_{gap} + 1$ more blocks than the altruistic main chain – but must achieve this within the upper bound of N blocks (maximum funded attack duration). Each new block is appended to the main chain with probability p_A , and to the attack chain with probability p_R respectively ($p_A + p_R = 1$). We, therefore, seek all possible series of blocks being appended to either chain and calculate the sum of the probabilities of the series which lead to a successful attack. In a successful series $i \in \mathbb{N}$ blocks are added to the main chain and $k_{gap} + i + 1$ blocks are added to the attack chain. The probability for such a series is $p_R^{k_{gap}+i+1} \cdot p_A^i$.

For any prefix strictly shorter than the whole series, the number of appended blocks to the attack chain is smaller than $k_{gap} + 1$, as otherwise, the attack would have ended sooner. It follows that the last block in a successful series is always appended to the attack chain. The number of combinations for such a series is derived similarly to Bertrand's ballot theorem, with a difference of k_{gap} for the starting point:

$$combinations(i) := \binom{k_{gap} + 2i}{i} - \binom{k_{gap} + 2i}{i - 1} \quad (4.4)$$

Assuming the attacker can only fund up to N blocks on the attack chain, the probability of a successful attack is hence given by:

$$\sum_{i=0}^{i \leq N - k_{gap} - 1} combinations(i) \cdot p_R^{k_{gap}+i+1} \cdot p_A^i \quad (4.5)$$

Using the formula 4.5 we can calculate the success probability of the attack. Figure 4.2 shows the probabilities for different values of rational hashrate p_R , as well as different amounts of blocks N these bribed miners can be rewarded/compensated for. The number of confirmation blocks required by victim Vincent is $k_V = 6$. Clearly, the attack requires $N > k_V$ to have a chance of being successful. As with the classical 51% attacks, the

¹²Another explanation can be that some miners have imperfect information, which might be the case in practice.

4. PAY TO WIN ATTACKS

Rational hashrate p_R	Average whale attack costs epoch reward 12.5 $cost_{whale}$ in BTC	P2W epoch reward 12.5 $cost_{exp.}$ in BTC	P2W cost compared to whale	P2W N average	P2W epoch reward 6.25 $cost_{exp.}$ in BTC
0.532	293e+23	196.50	$\approx 0.00\%$	109	159.00
0.670	999.79	108.50	10.85%	21	71.00
0.764	768.09	101.50	13.21%	14	64.00
0.828	1265.14	98.50	7.79%	11	61.00
0.887	1205.00	96.50	8.01%	9	59.00
0.931	1806.67	96.50	5.34%	9	59.00
0.968	2178.58	95.50	4.38%	8	58.00
0.999	2598.64	95.50	3.67%	8	58.00

Table 4.1: Comparison of attack costs for $k_V = 6$, all costs given in BTC. The costs for the whale attack are the average from 10^6 simulation results provided in [LK17]. For comparison different Bitcoin block reward epochs (12.5 and 6.25 BTC) are provided for our P2W attack, all with $cost_{operational} = 0.5$ BTC, and average fee per block of 2 BTC and a bribe $\epsilon = 1$ BTC.

attack eventually succeeds once the bribable hashrate is above the 50% threshold and the number of payable blocks N grows.

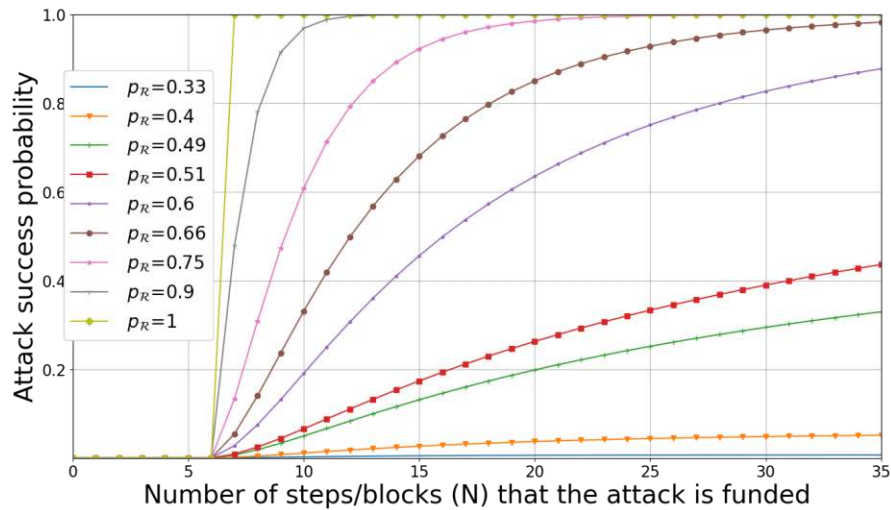


Figure 4.2: Attack success probability of a double-spending attack depending on the amount of blocks N that can be compensated/rewarded and different values for the rational hashrate p_R . The number of required confirmation blocks by Vincent is set to $k_V = 6$.

In other words, assuming more than $p_R > 0.5$ rational hashrate, bribing attacks are eventually successful if they can be funded long enough. The relevant question is how expensive it is to sustain the attack for a long enough period s.t., the attack is expected to be successful.

Table 4.1 shows a comparison between the expected costs of a successful P2W attack, against the average costs of 10^6 simulations of the whale attack as presented in [LK17]. At a first glance, given that the attacker must pay collaborating miners regardless of the

outcome of the attack, one may assume that the costs faced by the attacker are high compared to other bribing schemes. However, this is not the case. In our attack, miners face *no risk* from participation – requiring only a *low bribe value* to incentivize sufficient participation for a successful attack, contrary to existing bribing attacks like the whale attack.

It can be observed that, in contrast to the whale attack, our attack becomes cheaper when $p_{\mathcal{R}}$ grows large since the race is won faster and therefore fewer bribes have to be paid. Moreover, the whale attack has to pay substantially more funds to account for the risk rational miners face if the attack fails. Our approach is hence approximately $\approx 87\%$ to $\approx 96\%$ cheaper than the whale attack. For $p_{\mathcal{R}} = 0.532$ the difference is so large, that the costs of our P2W attack are insignificant compared to the whale attack. The switch to a new Bitcoin block reward epoch has further reduced the costs of the attack s.t., the costs of a successful double-spending attack ($k_V = 6$) using our technique are around 60 BTC. In October 2020 alone, there were around 60 thousand Bitcoin transactions with outputs greater than 60 BTC¹³.

4.4.4 Evaluation of Desynchronization

Publishing new block templates timely is a key requirement of this attack. So the question is, can we rely on the assumption that the difference between block intervals on the two chains, namely Bitcoin and Ethereum, is big enough such that before every new Bitcoin block there will be a new Ethereum block announcing the new block template? In other words: What is the probability that the two chains (funding and attack chain) *desynchronize* during an attack, i.e., that two Bitcoin blocks are mined in close succession without an Ethereum block in between. To identify the need to account for such events within the duration of an attack, we analyze the probability that the block intervals fluctuate in a way such that Bitcoin blocks are mined in close succession.

The time between Bitcoin and Ethereum blocks follows an exponential distribution. Assuming constant difficulty and overall hashrate, Ethereum has a mean block interval, i.e., an expected value of 15 seconds ($E_{ETH}(x) = 15$), whereas Bitcoin has a mean block interval of $10 \cdot 60$ seconds ($E_{BTC}(x) = 600$). To approximate the probability that the two chains desynchronize, we first calculate the probability that the time between two Bitcoin blocks is less than the Ethereum mean block interval ($x = 15$):

$$\lambda = \frac{1}{E_{BTC}(x)} \quad (4.6)$$

$$P(X < x) = 1 - e^{-\lambda \cdot x} \quad (4.7)$$

$$P(X < 15) \approx 2.47\% \quad (4.8)$$

The probability that this happens within N Bitcoin blocks i.e. the probability that the time between two Bitcoin blocks is smaller than 15 seconds during N total Bitcoin blocks

¹³c.f. [https://blockchair.com/bitcoin/outputs?s=value\(desc\),time\(desc\)&q=time\(2020-10\),value\(6000000000..\)#](https://blockchair.com/bitcoin/outputs?s=value(desc),time(desc)&q=time(2020-10),value(6000000000..)#)

is given by:

$$P(N) = 1 - (1 - P(X < 15))^{N-1} \quad (4.9)$$

$$P(32) \approx 53.93\% \quad (4.10)$$

This result already shows that it is necessary to provide templates for more than one Bitcoin block in one Ethereum block when executing long-running attacks.

We are now interested in the number of block templates the attacker has to provide per Ethereum block. Therefore, we analyze how probable it is that at least n Bitcoin blocks are mined before one Ethereum block. We approximate this value by calculating the probability that at least n Bitcoin blocks are found within the Ethereum mean block interval of 15 seconds. The Bitcoin block discovery is a Poisson point process, where the Poisson distribution parameter $\Lambda = E(X = n) = \frac{t}{E_{BTC}(x)}$ refers to the expected value of the number of events happening within $t = 15$ time. Then the complementary probability of finding at most $n - 1$ blocks is given by:

$$P(X > n) = 1 - P(X \leq n - 1) \quad (4.11)$$

$$P(X \leq n) = F(x) = e^{-\lambda} \sum_{i=0}^{n-1} \frac{\lambda^i}{i!} \quad (4.12)$$

$$P(X > 1) \approx 2.47\% \quad (4.13)$$

$$P(X > 2) \approx 0.03\% \quad (4.14)$$

$$P(X > 3) \approx 2.556 \cdot 10^{-4}\% \quad (4.15)$$

$$P(X > 4) \approx 1.595 \cdot 10^{-6}\% \quad (4.16)$$

Since both chains start at the same point in time, $n = 1$ already refers to a sequence of two Bitcoin blocks without an Ethereum block in between. We now calculate the probability that at least n Bitcoin blocks are found within the mean Ethereum block interval t during a period of N Bitcoin blocks in total:

$$P(n, N) = 1 - (1 - P(N > n))^{[(N-1)/n]} \quad (4.17)$$

$$P(n = 1, N = 32) \approx 53.930\% \quad (4.18)$$

$$P(n = 2, N = 32) \approx 0.490\% \quad (4.19)$$

$$P(n = 3, N = 32) \approx 0.003\% \quad (4.20)$$

So when providing three Bitcoin block templates, there remains approximately a 0.490% chance that all of them are consumed before the next Ethereum block is published.

To further justify these numbers and account for the fact that Ethereum blocks are exponentially distributed as well, we implemented a tool to simulate such parallel blockchain chain executions. Measuring the probability of desynchronization yields comparable results to our calculations with a mean Ethereum block interval of 15 seconds. After 10,000 runs of our simulation limited to $N = 32$ total Bitcoin blocks each, a chain

of at least two consecutive Bitcoin blocks before a corresponding Ethereum block was found in 53.0% of all cases. A chain of at least three consecutive Bitcoin blocks was found in 1.57% of all cases, a chain of at least four consecutive Bitcoin blocks in 0.08% of all cases. Consecutive chains of length 5 or longer have never occurred during 10,000 runs.

Desynchronization Prevention

As Section 4.4.4 shows, the attacker should not rely on the assumption that the difference between block intervals on the two chains, e.g., Bitcoin and Ethereum, is big enough such that before every new Bitcoin block there will be a new Ethereum block announcing the new block template. Therefore, the attacker is advised to publish block templates for multiple blocks in advance (leaving references to previous blocks to be filled in by miners)¹⁴. In this case, only the first block includes a *previous block hash* field, whereas in subsequent block templates this value is left empty and has to be filled by collaborating miners based on the current attack chain state. Later, the contract can use previously submitted valid attack blocks to check the validity of the submitted solutions, i.e., if they form a valid chain and have sufficient difficulty. This solution is implemented in our PoC.

Other approaches to ensure that new block templates are available to rational miners independently of block intervals in the funding cryptocurrency are also conceivable. Strictly speaking, it is not even necessary that the Ethereum block with the new block template has been mined before the next Bitcoin block for which the template has to be used. This is possible if the attack contract is implemented such that it enables collaborators to provide a valid Ethereum transaction signed by the attacker as proof that the therein announced new block template for a specific attack was approved, alongside their solution. Then any such transaction can be seen as a guarantee for the collaborating miners that they will receive a reward if they mine a block according to the template. At some later point, the transaction defining the target chain block template is included in the funding cryptocurrency and presents proof to the attack contract that the respective block on the target chain was based on a valid template. The drawback of this method is, that it requires some way to prevent equivocation of the attack operator to prevent that more signed block templates are available than actual funds in the contract, or alternatively more funds in the attack contract as a safety margin.

4.4.5 Evaluation of the attack contract

To verify the outcome of the attack and correctly pay rewards in trustless out-of-band scenarios, our attack contract includes a construction which we call an *ephemeral mining relay* (EMR)¹⁵, as it combines the functionality of a chain relay [But16, ZHL⁺19, Rel] and mining pool [VTL17, LVTS17].

¹⁴Furthermore, in practice collaborating miners would want to have at least a couple of block templates available to ensure that their hardware does not stall.

¹⁵We use the term “ephemeral” as the mining relay is instantiated only temporarily and does not require verification of the entire blockchain, but only the few blocks relevant for the attack.

Chain relays are smart contracts which allow to verify the state of other blockchains, i.e., verify the proof-of-work and difficulty adjustment mechanism, differentiate between the main chain and forks, and verify that a transaction was included within a specific block (via SPV Proofs [BCD⁺14]). However, a naïve chain relay implementation only allows to verify that a certain block (or transaction) was included in a chain with the most accumulated proof-of-work (i.e., heaviest chain). It does not allow to verify whether the blocks and transactions included in this heaviest chain are indeed *valid*, i.e., adhere to the consensus rules of the corresponding blockchain.

In contrast to previous proposals, our EMR needs to be capable of validating if blocks adhere to the consensus rules of the target cryptocurrency. This is achieved by restricting the allowed block structure. In our case, the set of transactions within blocks generated by collaborating miners is specified by the block template provided by the adversary. As Blofeld wants to submit collected PoW solutions to Bitcoin, it is in his best interest to provide only templates including valid transactions. Conversely, collaborating rational miners do not care if the block template they mine on is actually valid in Bitcoin, since the rewards they receive for solutions are guaranteed to be paid out by the smart contract in Ethereum.

Liveness

The liveness of chain relays in general depends upon the submission of new blocks to advance their state. Therefore, if the relay starves through a lack of submitted blocks, long-range attacks have a higher chance to succeed, as attackers gain additional time to compute long fake chains. In our concrete EMR instantiation liveness is less of an issue, as the duration of the attack is finite and well defined. Moreover, involved actors have an incentive to submit the correct information to the relay in a timely fashion. Consider, for example, a rational miner R who mined a block for the attack chain according to a template. Then R has an incentive to submit the solution to the PoW for this template timely since he is competing with other rational miners for the offered rewards and bribes. As the additional bribe ϵ is only paid if the attack is successful, this further incentivizes rational miners to publish solutions timely. Our scenario also enables the attacker, at any stage, to cease publishing additional block templates in order to reduce his losses in case the attack appears to fail.

Operational Costs

We implement a fully functional attack contract including the EMR on Ethereum, which is capable of verifying the state of the Bitcoin blockchain¹⁶. We use Solidity v0.6.2 and a local Ganache instance for cost analysis, with a gas price of 45 Gwei and an exchange rate 500 USD/ETH. The cost estimates for the identified operations are summarized in Table 4.2. Submitting a block template for a Bitcoin block amounts to 302,228 Gas (\$ 6.80 USD). The costs for submitting and verifying a new Bitcoin block are 468,273 Gas

¹⁶https://github.com/kernoelpanic/pay2win_artefacts

Operation	Approx. costs	
	Gas	USD
<i>Deployment</i>	6 156 688	138.53
<i>Initialization phase</i>	1 364 277	30.70
<i>Attack phase</i>	8 203 136	184.57
<i>Payout phase</i>	64 511	1.45
<i>Total operational costs</i>	15 788 612	355.24
<i>submit one block template</i>	302 228	6.80
<i>submit one block</i>	468 273	10.54

Gas price 45 Gwei, exchange rate 500 USD/ETH [Coi]

Table 4.2: Overview of operational costs $c_{operational}$ for each of the main Ethereum smart contract operations of the attack contract (including the EMR) executing a successful attack on Bitcoin with $k_V = 6$ and $k_B = 6$.

(\\$ 10.54 USD) in the worst case. In total, the costs of an example attack on Bitcoin with $k_V = 6$ and $k_B = 6$ are about \\$ 355.24 USD. This confirms that the costs for maintaining an attack contract including an EMR are marginal when compared to the potential scale of incentive attacks described in this chapter. For comparison: the reward for a single Bitcoin block (*excluding* transaction fees) at the time of writing is approximately \\$ 120 000 USD.

4.4.6 Ideas for Cost Optimizations

The biggest cost in the proposed out-of-band transaction revision attack derives from the compensation of k_V main chain blocks to provide an incentive for all rational miners (which already have contributed blocks to the main chain between block b_1 and b_{k_V}) to switch to the attack chain. In a blockchain where every block is uniquely attributable to a set of known miners, and where the overall hashrate of those miners can be adequately approximated, the payout of compensations can be further optimized in various ways. As an example, consider the scenario where a small miner, compared to the other miners, is lucky and mines several blocks within k_V . Then it may be cheaper for the attacker to exclude this miner from being eligible for compensation since it is unlikely that he will substantially contribute to the attack chain.

Crowdfunding and Multiple Attackers

Our attack from Section 4.4.1 also opens up the possibility to be crowdfunded. The simplest crowdfunding approach would be to allow donations as soon as the attack contract has been deployed. This method allows to collect funds but does not offer any guarantees for the backers.

Ideally, any solution which incentivizes multiple attackers to perform double-spending attacks concurrently would allow splitting the funds for the attack among collaborators. Thereby, multiple double-spends of low-value transactions by different parties could be made feasible if they together accumulate enough attack funds (r_{budget}). The main challenges that have to be solved in such a scenario are as follows:

- It has to be ensured that every collaborating attacker, who invests funds to achieve a double-spend attack, has some chance that his individual double-spend is successful, i.e., if the invested value is used by the contract, then the according double-spend attack for the respective transaction has to be performed.
- It has to be ensured that the attack cannot be poisoned by collaborating attackers such that they are able to sabotage the whole attack for all participants, i.e., it should not be possible for any participant to cause the attack to fail because of their inputs.
- The attack should not rely on any trusted third party.

We now outline an approach to achieve these goals within the framework of our previous attack s.t. its general structure is preserved. On a high level, the stages of the modified attack are as follows. First, the initialization transaction only announces that an attack might happen and that the block interval from b_1 to b_{k_V} will be affected. Then, all Bitcoin users who have performed transactions in block b_1 can decide whether or not to invest in the attack to potentially double-spend their transaction. The collaborating attackers, i.e., the backers, submit their double-spending transaction to the contract, together with some bribing funds in Ether that increase the overall funds r_{budget} of the attack¹⁷. This crowdfunding attack approach can be viewed as a practical instantiation of an analysis performed in [Bud18], where all payouts of a single block are viewed as the theoretical gain of a double-spending attack. In [Bud18] such a situation is analyzed from a financial perspective regarding the overall achievable economic security of Nakamoto consensus.

If the funding goal for reverting at least $k_V + 1$ blocks has been reached, the attack starts as previously described. Since the attacker who initialized the contract has to take care of producing new block templates for the chain containing the double-spend transactions, some method has to be implemented so that the transactions of other backers are assured to be included in b'_1 . We describe a method which requires a collateral from the original attacker (Blofeld) as high as the funds he wants to collect (r_{budget}). In doing so, it can be ensured that the other backers only pay if their transaction was actually included in the new chain in block b'_1 , which can be proven to the smart contract. Otherwise, they are refunded from the collateral submitted by the initial attacker.

The phases of the attack are as follows:

1) Blofeld who initiates the attack, deploys an attack contract in Ethereum and locks his collateral of value r_{budget} with this contract. Additionally, he publishes his spending transaction tx_{B_1} on the main network.

¹⁷Theoretically, an attacker can also specify a fixed rate of funds he wants to collect, depending on the overall value of the submitted Bitcoin transaction which should be double-spent.

- 2) Once k_V blocks on the main chain have been mined, Blofeld initializes the attack contract with his double-spend tx'_{B_1} , his part of the attack funds r_{B_1} , a reference to the block b_1 which is to be forked, as well as a reference to the common ancestor block b_0 .
- 3) Everybody who has included a transaction in block b_1 is then allowed to submit double-spending transactions $tx'_{B_{\{2,\dots,x\}}}$ including some amount of Ether $r_{B_{\{2,\dots,x\}}}$ that he or she is willing to invest in the attack. If these backers reach the funding goal of compensating at least $k_V + 1$ blocks before $k_V + 1$ main chain blocks have been submitted to the attack contract, the attack starts automatically. All invested funds (excluding the collateral r_{budget}) are then free to be used by the EMR as described in the original attack.
- 3) Once the attack has been started by the attack contract, Blofeld publishes a block template to the attack contract. The Merkle branch of this template includes all submitted double-spending transactions $tx'_{B_{\{2,\dots,x\}}}$, which are i) valid according to information from his full node ii) backed by some ether. Additionally, the attack contract has to require some freshness information such that Blofeld is unable to produce blocks before officially starting the attack to rip compensations increasing his invested value r_{B_1} from his fellow backers. An example of such a freshness guarantee would be the inclusion of the latest funding chain block hash e_1 in the block template.
- 4) Then the attack proceeds as originally described.
- 5) When N blocks are mined and published to the attack contract, the backers who have not witnessed that their double-spending transaction was included in the attack chain can now claim their invested Ether back from the attack contract. Therefore, the attack contract automatically allows any backer to reclaim their money if Blofeld cannot submit a valid Merkle inclusion proof for the respective double-spending transaction.

In this approach, Blofeld has to provide a collateral as large as the total funds required for a successful attack r_{budget} . If he behaves honestly, the collateral will be returned to him by the attack contract once the attack has ended – regardless if it was successful or failed. The collateral ensures that the initiator is able to compensate additional backers, in case their funds were used for the attack, but Blofeld did not include their double-spending transaction(s).

Like all other backers, Blofeld is required to invest funds r_{B_1} into the double-spending attack (in addition to the required collateral r_{budget}). This investment by Blofeld should ensure that he is indeed willing to execute an attack. At the same time, he also loses funds if he is not able to provide correct block templates. For example, if he as an initiator purposely stalls the attack e.g., by not producing any block templates, or not forwarding them in time to the Bitcoin main network, the attack will fail. But then he will also lose his invested funds r_{B_1} . Thereby, asymmetric losses in cases where Blofeld intentionally lets the attack fail can be avoided by backers. Thus, backers are advised not to invest more Ether than r_{B_1} provided by Blofeld.

Available Crowdfunding Funds

With the possibility to crowdfund attacks, theoretically, multiple double-spends of low-value transactions by different parties could also be made feasible if they together accumulate enough attack funds (r_{budget}). The discrepancy between the value transferred in one Bitcoin block and the rewards (including fees) distributed for mining one Bitcoin block show that the funds for long-range double-spending attacks using this technique are theoretically available. Over the last year (2019) the median value of bitcoins transacted per day (excluding change addresses) is approximately 780 million USD, whereas the median mining reward per day including transactions fees is approximately 11 million USD¹⁸.

4.5 P2W Transaction Exclusion and Ordering Attack

In this section, we describe a modification of our attack from Section 4.4.1, which exclusively targets transaction exclusion and ordering. Thereby, the attack becomes less powerful but also substantially cheaper, as we show in our evaluation. Nevertheless, the resulting attack could be used to perform multiple front-running attacks at once and censor certain transactions. Such attacks can be profitable for an attacker attempting to falsely close an off-chain payment channel (i.e., publish an old/invalid state) but prevent the victim from executing the usual penalizing measures [PD16, MBKM19, DW15]. The attack presented in this section can also be viewed as a form of the *feather forking* attack proposed by Miller [BMC⁺15]. In a feather fork, the attacker publicly promises that he will ignore any block containing a blacklisted transaction. The main attack proposed in this chapter uses smart contracts on a funding cryptocurrency to provide a more credible threat.

Initialization Phase (init)

The attacker's goal is to prevent an unconfirmed transaction tx_V from being included within N newly mined Bitcoin blocks. As in our previous attack, the smart contract is initialized and updated with *block templates*, which specify the content of the block according to the needs of the attacker. These templates have to be used by the collaborating Bitcoin miners to be eligible for rewards. This allows the attacker to fully control the content of the mined blocks, including ordering and inclusion of only desirable transactions. For each block template, the corresponding compensation and bribe is conditionally locked within the smart contract, ensuring miners will be reimbursed independently of the final attack outcome as long as they provide a valid solution. In contrast to our first attack, the attack can start immediately after it is initialized and does not have to wait for k_V blocks to pass, nor must it be initialized with $k_V + 1$ blocks.

¹⁸Numbers retrieved from <https://www.blockchain.com/charts>

Attack Phase (update)

As in our previous attack, rational miners submit valid Bitcoin blocks, based on the attacker's block templates, to the attack smart contract on Ethereum via the attack contract, which implements an EMR and verifies that they form a valid chain. At each step, the attacker can add new Bitcoin block templates after each submission to the attack contract and, if necessary, can even increase the bribes. If no new templates are submitted, the attack contract switches to the payout phase after k_B blocks. Note that it is possible to include more than one block template in a single block, as shown in Figure 4.3 for block e_3 (for details see Section 4.4.4).

Payout Phase (pay)

Miners can claim payouts in the attack contract once k_B Bitcoin blocks have been mined after the attack has ended (k_B being a security parameter defined by the attacker). The attack smart contract is responsible for verifying the validity of submitted blocks, i.e., their PoW in compliance with the specified block template, and that all blocks form a valid attack chain. If a submitted PoW is valid, the attack contract rewards miners even if the attack chain did not succeed to become the main chain, i.e., collaborating miners *face no risk*. The first miner to submit a valid PoW for the respective block template will, in any case, receive value equivalent to the full Bitcoin block reward in Ether, regardless if the attack has failed, plus an extra ϵ if the attack is successful. Thereby, it is important to define the success condition upfront (i.e., the number of subsequent compliant blocks) so that it is clear for bribees when the additional ϵ is payed.

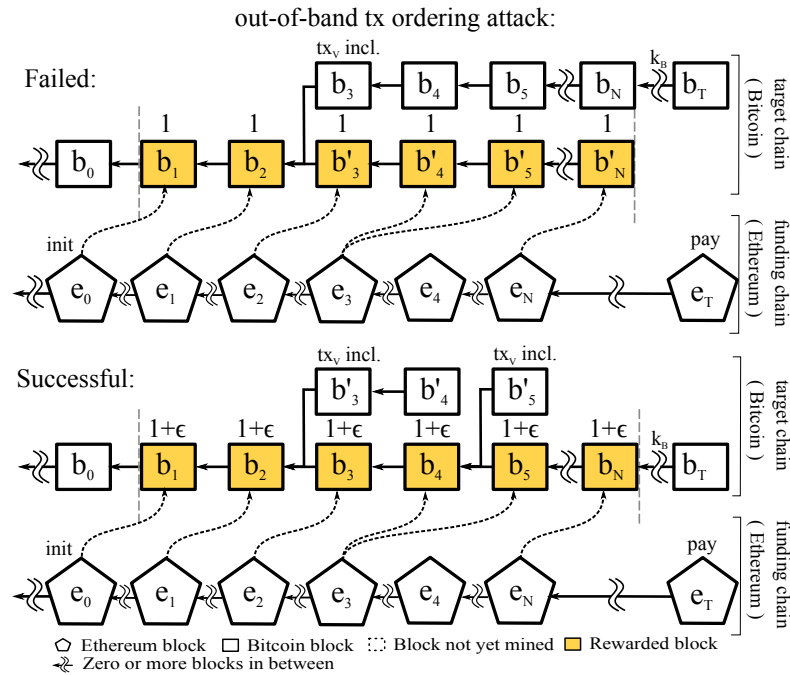


Figure 4.3: Example timeline showing blockchain structure and resulting payouts of a failed, and a successful transaction exclusion and ordering attack with out-of-band payments. The attack is initialized when the attack contract is published in block e_0 . Block templates are published as transactions in the funding cryptocurrency and refer to blocks in the target cryptocurrency. The payouts can be performed after k_B blocks. The colored blocks are rewarded by the attack contract, either only with their original value (reward + fee normalized to 1) or with an additional ϵ if the attack was successful. The numbers above colored blocks indicated those normalized rewards for the respective block.

4.5.1 Evaluation with Solely Rational Miners ($p_R = 1$)

We now derive a lower bound for the financial resources (*budget*) required from Blofeld in Ether (f_B) for this attack. Let us assume Blofeld wants to run the attack for N blocks.

Necessary Attack Budget

The budget of the attack contract must cover and compensate all lost rewards¹⁹, for every Bitcoin attack chain block in Ether in case the attack fails, plus an extra bribe ϵ per block in case the attack was successful. These values together with the funds of the attacker r_{budget} , define the maximum duration of the attack N in terms of attack chain

¹⁹This encompasses, block rewards including fees. In a concrete attack r_{reward} is not constant, but given by the coinbase output values of every submitted block.

blocks that can be financed:

$$r_{budget} = N \cdot (r_{reward} + \epsilon) + cost_{operational} \quad (4.21)$$

There, $cost_{operational}$ specifies the operational costs for smart contract deployment and execution (e.g., gas costs in Ethereum). Compared to the current block rewards, the operational costs for managing the smart contract are insignificant given the measurements in [MHM18] and Section 4.4.5. Although, costs currently being around 166 USD (see 4.4.5), we decided to set $cost_{operational} = 0.02$ BTC to provide a future-proof and permissive margin. Assume an attacker wants to specify the transaction ordering and/or exclusion in Bitcoin for the duration of one hour i.e., $N = 6$. A lower bound for the budget of the attacker r_{budget} can thus be derived by the current block reward (6.25 BTC) including approximated²⁰ fees (1 BTC) amounting to $r_{budget} = 7.25$ BTC. Providing an additional $\epsilon = 1$ BTC, yielding approximately 49.52 BTC as a lower bound for the budget in this example.

Costs of a Failed Attack

Although the attack cannot fail in a model where all miners are rational and the attacker has enough budget, it is relevant for a scenario where $p_{\mathcal{R}} < 1$ to determine the worst case cost for an unsuccessful attack. Note that the actual costs for a failed attack can be much lower, since Blofeld is able to halt the attack by not publishing any further block templates. In the *worst case* the attack duration is N and not one block produced by complacent miners (according to a published block template) made it into the main chain. Then the costs would be close to the maximum budget, reduced by $N \cdot \epsilon$, which amounts to approximately $cost_{fail} = 43.52$ BTC with our chosen values for N , r_{budget} and $cost_{operational}$.

Costs and Profitability of a Successful Attack

If the attack is successful, the attacker earns the block rewards on the main chain in BTC which compensate his payouts to bribed miners in Ether. The costs for a successful attack are thus reduced by $N \cdot r_{budget}$ main chain blocks, whereas rewards must be paid for $N \cdot (r_{budget} + \epsilon)$ block templates.

$$cost_{success} = N \cdot \epsilon + cost_{operational} \quad (4.22)$$

Therefore, the costs of a successful attack *only depend on the bribe paid per block as well as the operational costs*. Since we assume only rational miners, the attack in this scenario is always successful and *no-fork* will be required if $\epsilon > 0$. For a successful attack to be profitable, the amount gained from ordering, or transaction withholding, must exceed $cost_{success}$.

²⁰According to <https://blockchain.com/charts> the average transaction fees per Bitcoin block over the last year are 0.73 BTC. Accounting for standard deviation of fees and produced blocks per day the value varies between 0.79 BTC and 0.67 BTC. To provide a permissive margin we round to 1 BTC.

While the attacker must have the funds to compensate collaborating miners regardless of the outcome of the attack – the attack becomes cheaper than comparable attacks since the additional bribe does not have to account for the risk of getting nothing, faced by rational miners in other bribing scenarios. Other previously proposed AIM approaches require the attacker to have a sizeable portion of the overall hashrate (in the target cryptocurrency) under their direct control to stand a chance. For example `CensorshipCon` [MHM18] which is also aiming at transaction exclusion but specifically for Ethereum, requires $p_B > 1/3$, or `Script Puzzle 38.2%` as described in [TJS16], which would also allow to change the order of subsequent blocks requires $p_B > 38.2\%$. Acquiring or sustaining the required amount of hashrate already bears large costs, not to mention the additionally required bribes. The costs for renting 38.2% of Bitcoin’s total hashrate with NiceHash²¹ for the duration of one hour alone, amount to approximately 500 000 USD.

The `Pay per ...` attacks proposed in concurrent work [WHF19] operate in a comparable setting as our described attack and also highlight the economic feasibility without going into detail how such attacks can actually be constructed. The main differences to our attack are, that they focus on an in-band setting and only consider a model where all miners are rational.

4.5.2 Evaluation with Altruistic Miners

$$(p_A > 0 \wedge p_R + p_A = 1)$$

In a scenario where not all miners switch to the attack chain immediately, i.e., some of them act altruistically, some miners follow the protocol rules, but do not attempt to optimize their revenue, contrary to economically rational or bribable miners. Therefore, blocks of altruistic miners are likely to also include transactions and transaction orderings that are undesirable to the attacker. Therefore, blocks of such miners may have to be excluded by the attacker, i.e., by providing templates which intentionally fork away these blocks. If altruistic miners find a block, the attacker and colluding miners must mine at least two blocks for the attack chain to become the longest chain again – which altruistic miners will then follow. Hence, the security parameter k_{gap} is equal to 1 in this case, as we start our attack immediately after one undesired block has been mined. Therefore, *no* deep-forks of some length k_V (defined by the victim) are required in this scenario.

For a details analysis regarding costs and success probabilities of this case we refer to [JSSW22b], or Chapter 5 of this thesis. The resulting success probability of the attack has an influence on the choice of N and thus on the required budget r_{budget} , but the calculations for the respective bounds in terms of costs are the same as in the previous model with only rational miners (Section 4.5.1).

²¹cf. <https://www.crypto51.app/>

4.6 Discussion and Mitigations

Our AIM attacks serve to highlight the security dependency between transaction value and confirmation time k_V , as also stated in [SZ16]. As with the negative-fee mining pools presented by Bonneau in [Bon16], there exists an interesting analogy between such an incentive manipulation attack and a mining pool. At an abstract level, the presented attacks rely on a construction comparable to a mining pool, where the pool owner/attack operator defines specific rules for block creation for the targeted cryptocurrency within a smart contract. Moreover, every participant must be able to claim their promised rewards in a trustless fashion, based on the submitted blocks and state of the targeted cryptocurrency. The construction of an *ephemeral mining relay*, presented within this chapter, provides exactly this functionality. Luu et al. [LVTS17] also proposes a mining pool (Smart pool) which itself is governed by a smart contract. However, its design and intended application scenarios did not consider use cases with malicious intent. Smart pool does not enforce any properties regarding the content and validity of submitted blocks beyond a valid PoW, as the intrinsic incentive among participants is assumed to earn mining rewards in the target cryptocurrency, which is only possible if valid blocks have been created.

The natural questions which arise from the presented attacks are: How likely are such AIM attacks to occur in practice? Can they be efficiently mitigated? We now discuss these questions and provide some directions to explore possible counter measures and limitations of the described attacks.

4.6.1 Practical possibility

The focus of this chapter is to improve upon existing attacks and demonstrate the technical feasibility of advanced bribing attack, as well as to evaluate the associated costs. Hereby, the long term interests of miners of course also play an important role. There may be scenarios where miners are capable of providing PoW for a target blockchain, but at the same time do not have any long-term interest in the well-being of the target. Consider the real-world example of Bitcoin and Bitcoin Cash which utilize the same form of PoW and can be considered competitors. Thus, the question if the proposed attacks are possible in practice is difficult to answer scientifically. There is already empirical evidence from previous large-scale attacks by miners, e.g., 51% attacks on Ethereum Classic [coi20, coi19b] Bitcoin Gold [btg20] Bitcoin Cash [coi19a], as well as incentive manipulation attacks, e.g., Fomo3D [fom18] and front-running [DGK⁺20]. To the best of our knowledge, none of the observed attacks has been as sophisticated as the new technique proposed in this chapter, but of course, attacks get better over time. Nevertheless, these cases demonstrate that large-scale attacks happen, and that the topic of incentives in cryptocurrencies is an area which deserves further study. We see our attack descriptions as another important contribution in this direction.

4.6.2 Counter-attacks

Counter-bribing refers to the technique of countering bribing attacks with other bribing attacks [Bon16, Bon18]. For the victim(s), counter-bribing is a viable strategy against AIM. The difficulty of successfully executing counter-bribing highly depends on the respective scenario. In the end, counter-bribing can also be countered by counter-counter-bribing and so forth. Therefore, as soon as this route is taken, the result becomes a bidding game. Against transaction exclusion attacks, counter-bribing can be performed by increasing the fee of tx_V such that it surpasses the value promised for not including the transaction²². If defenders have imperfect information, they may not be able to immediately respond with counter-bribes. In this case, some of the attack chain blocks may have already been mined, or even take the lead, before they are recognized by defenders. Counter-bribing then necessitates the incentivization of a fork, and thus a more expensive transaction revision attack, leading to asymmetric costs in the bidding game. This illustrates an important aspect of AIM, namely their visibility. On the one hand, sufficiently many rational miners of the target cryptocurrency have to recognize that an attack is occurring, otherwise they won't join in and the attack is likely to fail. On the other hand, if the victims of the attack recognize its existence, they can initiate and coordinate a counter-bribing attack. So the optimal conditions for AIM arise if all rational miners have been informed directly about the attack, while all victims/merchants do not monitor the chain to check if an attack is going on and are not miners themselves.

Although our proposed attacks are clearly visible on the funding cryptocurrency, they are not necessarily observable in the target cryptocurrency. In the best case, the proposed transaction exclusion and ordering attack does not even produce a fork in the target cryptocurrency when all miners act rationally. Even if forks are induced, participating miners can make use of the fact that the PoW mining process does not require any strong identity by using different payout addresses. Of course their received rewards can be traced in the funding cryptocurrency, but available privacy techniques may be used to camouflage the real recipient of the funds e.g., [MM17, HBG16].

The great benefit of the herein described attacks is that bribes are paid out-of-band. Hereby, our attacks are rendered more stealthy to victims, who only monitor the target cryptocurrency. It can hence be argued that counter-attacks by victims are harder to execute as they are not immediately aware of the bribing value that is being bet against them on a different funding cryptocurrency. We also follow the argument in [Bon16] that requiring clients to monitor the chain and actively engage in counter-bribing is undesirable, and our out-of-band attacks further amplifies this problem as clients would have to concurrently monitor a variety of cryptocurrencies.

Another interesting aspect of counter-bribing is revealed if crowdfunded attacks are assumed. In this case, the funds required to counter-bribe can be higher than the invested funds of each individual attacker. In a scenario with multiple victims, organizing

²²Another possible counter-attack would be to launch a DoS attack against the censor, see Section 4.6.2 for details

coordinated counter-bribing is difficult. All victims would be better off if the attack fails, but for an individual victim it is cheaper to not take action and hope that others will fund the counter-bribe, leading to a *collective action problem*.

DoS Against Transaction Censorship

We consider Bitcoin as a target, however in principle our transaction censorship attack is also applicable to other types of cryptocurrencies. Although, we argue that (quasi) Turing complete smart contract capable cryptocurrencies are more resistant to censorship than Bitcoin:

Let's assume, for the remainder of this discussion, that transaction censorship should take place within Ethereum as a target cryptocurrency. Moreover, the respective transactions, or their side effects, can be accurately identified and all miners agree that these transactions exhibit unwanted behavior. This opens up the possibility of denial-of-service attacks launched by the victim(s) in such a case. The reason for this stems from the fact that the effects of an unwanted transaction can be proxied through multiple layers of smart contract invocations and interactions. Hereby, the problem arises that miners may only learn of the unwanted behavior of a transaction by first evaluating its state changes. If the resulting behavior is to be censored, miners have to roll back all changes and cannot collect transaction fees for their efforts. Therefore, the attacker can waste the resources of every censoring miner without a loss of funds.

It is impossible to directly overcome this issue without changing the consensus rules, however by basing the attack on block templates, the problem is shifted away from the collaborating rational miners toward the attacker. Hereby, the attacker may choose to only include simple transactions for which he is certain that they cannot hide any unwanted activity e.g., all value transfer transactions, or calls to known contracts such as ERC20 Tokens.

Cross-chain Verifiability

One crucial aspect of our attacks is that a smart contract within the funding cryptocurrency must be able to validate core protocol and consensus rules of the target chain, in particular it must be able to determine the validity of blocks. If this is not possible, the attack cannot be executed trustlessly. For example, it is currently not possible to execute an AIM against Litecoin using Ethereum as a funding cryptocurrency in a fully trustless manner, as it is economically unfeasible to verify the Script hash function within a smart contract. However, it is generally beyond the reach of an individual cryptocurrency to dictate or enforce what other cryptocurrencies support in future versions of their smart contract languages. Thus, any such defensive decision of the target cryptocurrency may be mitigated by future changes in another cryptocurrency. Hence, such measures can not guarantee lasting protection.

On a high level the technical requirements which would allow to trustlessly execute our AIM attack using smart contracts can be generalized by the following points:

1. Given a block in a block interval (on the target chain) defined by the attacker, the smart contract must be able to verify either that a certain state transition was performed (e.g., a transaction was included in the blockchain), or that a certain state transition was not performed (e.g., a transaction was not included). In either way, the current state of the target cryptocurrency has to be determined after T blocks have been mined on top of a block pre-defined by the attacker, i.e., the longest chain. This implies that it is possible to verify the PoW of the target cryptocurrency on the funding cryptocurrency in a smart contract. To also be able to compensate block contributed to the attack, even if the overall attack is not successful, the state and length of the longest attack chain has to be determined as well.
2. A programmatic way to uniquely attribute blocks on the target cryptocurrency to miner addresses, as well as a way to map the latter to corresponding addresses in the funding cryptocurrency.
3. A programmatic way to transfer value in the funding cryptocurrency to a uniquely attributed address of a collaborating miner (see point 1).

4.7 Implications of P2W and AIM

In this chapter we introduced a new AIM attack method called Pay-To-Win (P2W) and showed that attacks utilizing the described techniques can readily be constructed given current smart contract platforms and are economically feasible in practice. The implications of our proposed method (and related AIM attacks) regarding the security guarantees of PoW cryptocurrencies are not yet conclusive and topic of future work. On the theoretical side, embedding and modeling incentive attacks in formalisms of Nakamoto style cryptocurrencies is non-trivial, as prevalent approaches do not consider rational participants [GKL15, PSS17, BMTZ17, GKR20], or explicitly exclude bribing [BGM⁺18]. Furthermore, no agreed upon game-theoretic analysis technique for (PoW) cryptocurrencies currently exists, and it remains an open question if such an analysis could be rendered universally composable. The generalization and inclusion of AIM attacks and rational behavior in formal analysis frameworks for Nakamoto consensus based cryptocurrency designs, including approaches such as *Proof-of-Stake*, hence poses an interesting and important open research challenge. On the practical side, our new attacks, as well as the existing body of research on AIM, demonstrates that it is not only the hashrate distribution among permissionless PoW-based cryptocurrencies that plays a central role in defining their underlying security guarantees. The ratio of *rational* miners and available funds for performing AIM also form a key component, as rational miners can be incentivized to act as accomplices to an attacker. The possibility of trustless out-of-band attacks highlights that being able to cryptographically interlink cryptocurrencies increases this attack surface. Further, smart contract based AIM introduces the possibility to align the interests of multiple attackers who want to perform double-spends during the same

time period, making low value double-spends theoretically feasible (as economically analyzed in [Bud18]). Together with the topic of counter-bribing, new research directions are opened up that raise fundamental questions on the incentive compatibility of Nakamoto consensus. Real-world attacks targeting incentives, such as front-running [DGK⁺20], demonstrate that the existence of incentives cannot be ignored in PoW cryptocurrencies, i.e., by only considering honest and malicious (Byzantine) miners. To accurately reflect the security properties of permissionless PoW cryptocurrencies, some form of rationality has to be taken into account. The problem is, that as soon as rational players are considered, all previously proposed AIM methods, as well as the attacks described in this chapter, lead to interesting questions whether or not the incentive structures of prevalent cryptocurrencies actually encourage desirable outcomes. Even more so, in a world where multiple cryptocurrencies coexist it is likely not sufficient to model them individually as closed and independent systems.

How much is the fork?

In this chapter¹, new techniques for estimating the success probability as well as the profitability of forks are proposed. Estimating the probability, as well as the profitability, of different attacks is of utmost importance when assessing the security and stability of prevalent cryptocurrencies. Previous modeling attempts of classic chain-racing attacks have different drawbacks: they either focus on theoretical scenarios such as infinite attack durations, do not account for already contributed blocks, assume honest victims which immediately stop extending their chain as soon as it falls behind, or rely on computationally heavy approaches which render them ill-suited when fast decisions are required. In this chapter, we present a simple yet practical model to calculate the success probability of finite attacks, while considering already contributed blocks and victims that do not give up easily. Hereby, we introduce a more fine grained distinction between different actor types and the sides they take during an attack. The presented model simplifies assessing the profitability of forks in practical settings, while also enabling fast and more accurate estimations of the economic security grantees in certain scenarios. By applying and testing our model in the context of bribing attacks, we further emphasize that approaches where the attacker compensates already contributed attack-chain blocks are particularly cheap. Better and more realistic attack models also help to spot and explain certain events observed in the empirical analysis of cryptocurrencies, or provide valuable directions for future studies. For better reproducibility and to foster further research in this area, all source code, artefacts and calculations are made available on GitHub.

5.1 Calculating the Success Probability of Forks

Whenever a miner in a permissionless Proof-of-Work (PoW) cryptocurrency detects a fork, this miner has to make a decision which chain to extend, i.e, where to best utilize

¹This chapter represents an extended version of publication [JSSW22b].

her hashrate. As a profit-oriented and economically rational miner, she would want to select the chain which offers the higher expected profit for the next block as fast as possible. Optimizing for maximum profit, other revenue opportunities besides the block reward of the next block, such as a *bribe* [Bon16], or some other way to increase the miner extractable value (MEV) [DGK⁺20, EMC19, QZG22], also have to be taken into account. We present and apply a simple model, that is precisely tailored towards the question of selecting the most profitable branch of a fork, as well as assessing the probability of this branch becoming canonical.

Modeling the security of PoW cryptocurrencies has been addressed from several different angles. One of the first approaches by Rosenfeld [Ros14] highlights that successful double-spending is possible with any attacker hashrate (no majority is needed), assuming that all non-attacking nodes are honest and accept the attack chain as soon as it becomes the longest/heaviest chain. Thereby, an infinite attack duration is implicitly assumed from the perspective of the attacker. Moreover, potential incentives of participants are ignored. Liao and Katz [LK17] extend the analysis from Rosenfeld [Ros14] in the context of chain forks incentivized by whale transactions, i.e., transactions which carry an exceptionally high fee which can be viewed as a bribe. Their model is simple and specifically tailored for bribing, but does not account for already contributed blocks, finite attacks, and victims that do not accept longer chains immediately. In this chapter, we build upon the model of Liao and Katz [LK17] and extend it through finite Markov chains. In context of cryptocurrencies, Markov chains have mainly been used to model selfish-mining [ES14, NKMS16], however, to the best of our knowledge, not directly to model double-spending or bribing attacks.

Markov decision processes (MDP) were successfully used in the past to model certain security aspects of PoW cryptocurrencies such as selfish-mining [SSZ16, GKW⁺16], or double-spending [GKW⁺16, ZQC⁺21]. The double-spending MDP in [GKW⁺16] is quite versatile and incorporates a lot of parameters (for example stale block rate and network connectivity). On the down side, it assumes that the *exit* state is reached as soon as the adversarial chain is ahead of the main chain. This assumption though may not hold in practice, in particular when economically rational victims are considered. Moreover, adapting existing MDP based approaches to account for already contributed blocks to the fork/main chain regarding the profitability is not straight forward, as it would require a substantial change in the design of currently available MDPs. We avoid this issue, by using a Markov chain solely to calculate the required probabilities and embed these in a formula that accounts for already contributed blocks to each chain. Last but not least, evaluating finite complex MDPs and applying binary search to find the maximum reward is more time consuming than evaluating a finite Markov chain. As the run time heavily depends on the chosen model and concrete parameterization, an accurate comparison is of course impossible if the underlying model is not exactly the same, but to give the reader some intuition we provide an approximation assuming a current desktop computer as underlying hardware: in this case, a broadly used MDP [GKW⁺16, ZQC⁺21] for finding optimal strategies has a runtime in the range of multiple minutes, whereas our

approach provides results within milliseconds, parameterized for practical fork ranges of around 6 blocks. Even when parameterized for forks that are multiple thousand blocks long, our approach still provides results within seconds. This is possible since the required calculations consist mainly of matrix multiplications and closed form formulas. In summary, our approach complements established MDPs used to find optimal strategies, as it does not incorporate network latency or stale block rate, but it provides a practically oriented and quickly computable model that also takes economically rational victims and already contributed blocks into account. Therefore, our approach poses a viable alternative, especially for scenarios where the overhead imposed by MDPs is undesired or unacceptable. All source code as well as all generated artefacts can be found on GitHub².

Also from an empirical analysis point of view, better models for attacks on prevalent cryptocurrencies are helpful, as they might offer explanations for observed fork patterns, or provide valuable directions for future studies and measurements which aim to detect forking patterns related to malicious activities of miners.

5.1.1 Related Work

As we discussed previously in the beginning of this introduction, in the context of cryptocurrencies, Markov chains have mainly been used to model selfish-mining [ES14, NKMS16]. More complex, Markov decision processes have been used to model selfish-mining [SSZ16, GKW⁺16] as well as double-spending [GKW⁺16, ZQC⁺21]. Closest to our approach is the line of research regarding the analysis of double-spending starting with Rosenfeld [Ros14], which was extended by Liao and Katz [LK17] to incorporate a basic notion of bribing/incentives. This is also the model which we extend upon in this chapter.

Lately, a series of works has empirically analyzed and automated the discovery and possible exploitation of MEV opportunities [DGK⁺20, ZQT⁺20, TIGS21, ZQG21]. Our work is orthogonal to this line of research and may best be compared by elaborating on the question whether or not to join/or initiate a blockchain fork to hunt a missed MEV opportunity. At the end of Zhou et al. [ZQC⁺21] this question has also been raised and was briefly addressed using the MDP from Gervais et al. [GKW⁺16], to derive thresholds for the required minimum MEV value and hashrate required to justify a fork. Our work can be seen as an extension to this question from a different angle, without the previously mentioned drawbacks of an MDP based approach, while accounting for already contributed blocks to a fork and economically rational victims.

5.1.2 Structure of this Chapter

In this chapter, we aim to model and analyze a range of different attacks from the perspective of an individual miner. These include classical longest chain races [Ros14], but also bribing [Bon16] and other attacks involving additional income, such as MEV opportunities [DGK⁺20, ZQC⁺21] and algorithmic incentive manipulation attacks [JSZ⁺21b].

²https://github.com/kernoelpanic/howmuchisthefork_artefacts

In Section 5.2 we first extend the system model provided by Judmayer et al. [JSZ⁺21b, JSZ⁺19], for example by allowing the victims to have hashrate. In Section 5.3 we build up and extend the calculation approach presented by Liao and Katz [LK17]. In Section 5.3.1 we first describe their original approach using our newly introduced notation from Section 5.2. We consider already contributed blocks in Section 5.3.2, economically rational victims as well as finite attacks with different pain thresholds for actors in Section 5.3.3 and economically incentivized attacks with effort-related compensation approaches in Section 5.3.4.

5.2 Roles, Types and Sides of Players

In this work, the focus of our analysis is solely on miners, nevertheless to increase the extensibility of our model and to offer a definition for further discussions, we differentiate between two player *roles* (*miner* and *user*) and three player *types* (*altruistic*, *Byzantine* and *rational*). Each player must have *one* role and at the same time fall into *one* of three types of actors³. In general, roles define the capabilities of a player, whereas player types define their overall strategic behavior. Additionally, during an attack the actors of the type *rational player* can be on one of three *sides* (*extractors*, *victim* and *indifferent*). On which side a rational player will be during an attack depends on various factors which influence the value at stake for the respective player, for example already contributed blocks to a chain. The set of all players is denoted \mathcal{P} and the number of players is $|\mathcal{P}|$. The sets of players are denoted in calligraphic letters, e.g., $\mathcal{A}, \mathcal{B}, \mathcal{R}, \mathcal{M}, \mathcal{U}$, where $\mathcal{P} = \mathcal{M} \cup \mathcal{U} = \mathcal{A} \cup \mathcal{B} \cup \mathcal{R}$.

5.2.1 Player Roles

The roles define the capabilities of a player. If player i controls some hashrate p_i , where $\sum_{i=1}^{|\mathcal{P}|} p_i = 1$, he is part of the set of miners \mathcal{M} and thus termed a miner. The number of miners in \mathcal{P} is denoted with $|\mathcal{M}|$. If a player does not control any hashrate, he is part of the set of users \mathcal{U} and thus termed a user. The number of users in \mathcal{P} is denoted with $|\mathcal{U}|$. It holds that $\sum_{i=1}^{|\mathcal{M}|} p_i = 1$, whereas $\sum_{i=1}^{|\mathcal{U}|} p_i = 0$.

5.2.2 Player Types

The different types define the general strategic behavior of a player. For our analysis, we uniquely assign a player $i \in \mathcal{P}$ to one of three mutually disjoint actor types, s.t. $\mathcal{P} = \mathcal{A} \cup \mathcal{B} \cup \mathcal{R}$.

Altruistic players (\mathcal{A}): The set of players which act altruistically. They always follow the rules of the protocol and hence do not deviate even if this would offer higher profits. The accumulated hashrate of these players is denoted by $p_{\mathcal{A}}$.

³In this work, the word actor and player are used interchangeably.

Byzantine players (\mathcal{B}): The set of players or a single player, which acts Byzantine and thereby in the most destructive way possible e.g., by executing an attack. The accumulated hashrate of these players is denoted by $p_{\mathcal{B}}$. For most of our attack analysis in this chapter we set $p_{\mathcal{B}} = 0$.

Rational players (\mathcal{R}): The set of players which act rational under economic considerations. They follow the rules of the initial protocol as long as there is no other strategy which yields higher profits. The accumulated hashrate of these players is denoted by $p_{\mathcal{R}} = 1 - (p_{\mathcal{A}} + p_{\mathcal{B}})$. The number of players that are rational is denoted by $|\mathcal{R}|$, where each player controls some fraction of the total hashrate s.t. $p_{\mathcal{R}} = \sum_{i=1}^{|\mathcal{R}|} p_{\mathcal{R}_i}$ holds. An example for a rational actor is Rachel (R). Rational players are the only type of player which can be further divided into different *sides* during an attack (see 5.2.3). Rationality always depends on a certain optimization criteria in question, i.e., the parameter which should be optimized by acting rational. In this chapter the rationally criteria is *short-term* maximization of funds, under the assumption that the exchange rate remains constant, analogous to most analysis approaches [LK17]. So the question rational players face is: “What is the next best block?”, i.e., the most profitable chain to extend.

5.2.3 Player Sides

During an attack a rational player can also be categorized depending on the side she is taking in the respective attack. This is only relevant for rational players, as they might change their strategy if this promises higher profits, while altruistic players for example never change their strategy. At any point, it holds that $\mathcal{R} = \mathcal{E} \cup \mathcal{V} \cup \mathcal{I}$.

Extractor(s)/ Exploiter(s) (\mathcal{E}): The set of players which does not follow the prescribed rules of the protocol to gain a financial advantage. They seek to exploit an additional value extraction opportunity, such as a front-running, arbitrage, a censorship, double-spend, or any other attack vector to increase their MEV beyond what they would get by block rewards and fees. Therefore, they might be willing to share some of their reward in order to bribe other miners, to aid in their attack. This set of players has hashrate $p_{\mathcal{E}}$. They join (or are assumed to join) an attack to gain a profit.

Victim(s) (\mathcal{V}): The set of players which would lose funds if a described attack is successful, e.g., a merchant who is the victim of a double-spend. This set of players has hashrate $p_{\mathcal{V}}$. Victims certainly work against an ongoing attack and even might launch a counter-attack if this is economically rational.

Indifferent (\mathcal{I}): The set of players which follows the prescribed rules of the protocol, although an attack is ongoing. This party does neither profit nor lose when the attack is successful. The difference between these actors and altruistic miners is that these indifferent rational miners would change their strategy as soon as they are positively,

or negatively, affected by an attack. Therefore, as long as the situation for them does not change, they can also be modeled as being part of p_A , which was implicitly done in previous modeling approaches. In this work, we explicitly define the hashrate of these actors by p_I , s.t., $p_R = p_E + p_V + p_I$.

To illustrate that the separation of economically rational actors into the mentioned sides is dependent on the viewpoint, we now provide an example.

5.2.4 Example: (Unindented) Fork

When a miner Alice (A) does not receive the latest block b_{n_B} from miner Bob (B) timely, she keeps on trying to extend an old block b_{n-1} . If now Alice finds a block b'_{n_A} and publishes it, this results in a unindented fork where Bob would build up on his block b_{n_B} and Alice would continue to mine on her block b'_{n_A} , even after receiving Bob's block for the same height. Although, such situations can be expected to happen during normal protocol operation, at that point Bob and Alice have no way to tell if this was a coincidence or happened because of malicious activity. Only if this pattern persists for a prolonged period of time, it would be possible to detect a skew in the distribution of blocks and the occurrence of forks, which would indicate malicious mining activity in retrospect. However, for a singular event this distinction between attack and normal operation is difficult.

Therefore, at that point Alice as well as Bob could also be viewed as adversarial from the perspective of the other party. From the perspective of Bob, Alice would be the aggressor, i.e., $A \in \mathcal{E}$, and he is the victim $B \in \mathcal{V}$, as he would lose his rewards from block b_{n_B} if b'_{n_A} becomes part of the longest chain. From the perspective of Alice, Bob would be the aggressor, i.e., $B \in \mathcal{E}$, and she is the victim $A \in \mathcal{V}$, as she would lose her funds from block b'_{n_A} if b_{n_B} becomes part of the longest chain. Under the assumption that the transactions in both blocks are equal, all other rational miners are indifferent from both viewpoints as they do not have any preference regarding one of the two blocks, i.e., $(\mathcal{R} \cap (A \cup B)) \in \mathcal{I}$.

If now a new block $b_{(n+1)_C}$ is found and published by Carol, which builds up on the block of Alice (b'_{n_A}) the situation changes again, depending on which type of Actor Bob is. If Bob is altruistic ($B \in \mathcal{A}$) he switches to the longer chain and the fork is over. If Bob is Byzantine then his actions can be modeled as the worst response without considering his individual losses or gains, i.e., in this case continue mining on top of b_{n_B} . If Bob is economically rational his choice which chain to adopt will depend on his expected profits and thus also on his hashrate and the resulting chance of winning the race. Also the set of indifferent miners changes: Carol now clearly has an interest in keeping the longest chain containing her block. Moreover, all other miners have a higher chance to gain rewards on the longest chain than on a shorter chain, therefore their expected reward will be higher on the longest chain as well.

5.3 Probability and Profitability Calculation

In this section we focus on the question which chain to extend in case of a fork, i.e., which next block provides the most expected profit. We address this question from the perspective of an individual rational miner m with hashrate p_m . Therefore, we build up on the model from Liao and Katz [LK17], which we first translate into the notations used in this work and then extend it. Thereby, we consider already contributed blocks, finite attack durations and individual pain thresholds of players, as well as economically rational victims.

5.3.1 Basic Model for Calculating Expected Profit

An inherent requirement for calculating the expected profit of the next block in any scenario where there is more than one chain that can be extended, is that the total hashrate which will work towards each individual chain has to be guessed, as also done in [LK17]. Considering the setting, where all players are economically rational ($p_{\mathcal{R}} = 1$), the miner m has to estimate $p_{\mathcal{E}}$, $p_{\mathcal{V}}$ and $p_{\mathcal{I}}$, in order to calculate the expected payoffs for the next blocks(s). Hereby, \mathcal{E} is assumed to work on the attack chain, and \mathcal{V} is assumed to work on the main chain, as these chains provide more individual profit for them. The players in \mathcal{I} are indifferent because they will neither lose nor profit from the attack and thus always work on the currently longest chain. A possible explanation for such a situation (considering imperfect information available to the parties) might be, that indifferent miners have not yet recognized that a profitable attack is going on. Another explanation would be, that their potential gains from the attack exactly cancel out their expected losses on the main chain.

The probability of an attack chain, or fork, to ever catch-up given an unlimited number of tries/blocks ($N = \infty$), if it is z blocks behind, was defined in [Nak08, Ros14] and is given by Equation 5.1.

$$\mathbb{P}(\text{succ. attack}) := \begin{cases} \left(\frac{\text{hashrate on fork chain}}{\text{hashrate on main chain}} \right)^{z+1} = \left(\frac{p}{1-p} \right)^{z+1} & \text{if } z \geq 0 \text{ and } p \leq 0.5 \\ 1 & \text{if } z < 0 \text{ or } p > 0.5 \end{cases} \quad (5.1)$$

In case of a fork/attack the miner m has two options: Either m *joins* the attack and extends the fork, or m *abstains* from the attack and thus extends the main chain. Depending on this decision, the resulting success probability of the attack changes as p_m is added to the hashrate working on the respective chain and thus against the other. The resulting success probabilities of the fork in both scenarios are as follows in our verbose

notation.

$$\mathbb{P}(\text{success join}) := \begin{cases} \left(\frac{p_{\mathcal{B}} + p_{\mathcal{E}} + p_m}{p_{\mathcal{A}} + p_{\mathcal{V}} + p_{\mathcal{I}}} \right)^{z+1} & \text{if } z \geq 0 \text{ and } p_{\mathcal{B}} + p_{\mathcal{E}} + p_m \leq 0.5 \\ 1 & \text{if } z < 0 \text{ or } p_{\mathcal{B}} + p_{\mathcal{E}} + p_m > 0.5 \end{cases} \quad (5.2)$$

$$\mathbb{P}(\text{success abstain}) := \begin{cases} \left(\frac{p_{\mathcal{B}} + p_{\mathcal{E}}}{p_{\mathcal{A}} + p_{\mathcal{V}} + p_{\mathcal{I}} + p_m} \right)^{z+1} & \text{if } z \geq 0 \text{ and } p_{\mathcal{B}} + p_{\mathcal{E}} \leq 0.5 \\ 1 & \text{if } z < 0 \text{ or } p_{\mathcal{B}} + p_{\mathcal{E}} > 0.5 \end{cases} \quad (5.3)$$

Since we aim to compare the profitability of mining on different chains of the same cryptocurrency, the costs for mining are the same regardless on which chain the hashrate is p_m is mining on. Therefore, as in [LK17] we can ignore the operational costs of mining in our calculation as it is the same on all chains. Moreover, we do not include the costs of acquiring the mining hardware in our calculation, as we assume that this has already been done and was based on an economically rational decision that mining in general can be executed profitably.

To now compare the expected profits of extending two different chains, we normalize the reward to 1 ($r_{block} = r_{blockreward} + r_{fee} = 1$) and assume that the exchange rate and thus the value gain from a block remains constant. Then the expected reward of m for one new block on the main chain if a fork/attack fails is given by Equation 5.4.

$$\rho_{main} := \frac{(1 - \mathbb{P}(\text{success abstain})) \cdot p_m}{p_{\mathcal{A}} + p_{\mathcal{V}} + p_{\mathcal{I}} + p_m} \quad (5.4)$$

Conversely, the expected reward of m for one new block on the attack chain if a fork/attack succeeds is given by Equation 5.5.

$$\rho_{fork} := \frac{\mathbb{P}(\text{success join}) \cdot p_m}{p_{\mathcal{B}} + p_{\mathcal{E}} + p_m} \cdot (\epsilon + 1) \quad (5.5)$$

Equation 5.5 already contains a potential bribe ϵ , which is paid out on per block basis in the competing attack chain. Thereby, also the expected profit for miners participating in bribing [Bon16] or algorithmic incentive manipulation [JSZ⁺21b] attacks can be modeled. If an unintentional fork without any bribes should be modeled, ϵ can be set to zero.

To derive the required bribe ϵ , as done in [LK17], the expected reward on the fork/attack chain has to be larger than the expected reward on the main chain. This means Equation 5.5 has to be larger than 5.4, i.e., $\rho_{fork} > \rho_{main}$. Rearranging this inequality yields.

$$\epsilon > \frac{\left(1 - \left(\frac{p_{\mathcal{B}} + p_{\mathcal{E}}}{p_{\mathcal{A}} + p_{\mathcal{V}} + p_{\mathcal{I}} + p_m}\right)^{z+1}\right)}{p_{\mathcal{A}} + p_{\mathcal{V}} + p_{\mathcal{I}} + p_m} \cdot \frac{p_{\mathcal{B}} + p_{\mathcal{E}} + p_m}{\left(\frac{p_{\mathcal{B}} + p_{\mathcal{E}} + p_m}{p_{\mathcal{A}} + p_{\mathcal{V}} + p_{\mathcal{I}}}\right)^{z+1}} - 1 \quad (5.6)$$

A briber can use this formula to estimate the required amount of the bribe to convince other economically rational miners to join the attack.

When analyzing the case of an infinite attack duration (i.e., Equation 5.1) an interesting observation regarding an artefact of this approach can be made. If the hashrate of the attacker exceeds $0.381966\dots$, then in forks of length one ($z = 1$) the expected profitability of the next block while staying on the fork is higher than it would be on the main chain i.e., in the case where the equivalent hashrate participates in honest mining. This particular hashrate value already occurred in some publications [TJS16, MHM18] in the context of cryptocurrencies, but was never discussed in detail. We now outline in Section 5.3.1 why this bound in hashrate of exactly $1/\phi^2$ exists and where it comes from.

The $1/\phi^2$ Bound

If no-fork, or attack, occurs the hashrate p_m of a miner and thus the probability to find a new block, is directly correlated with the expected profit for a new block (if normalized to one). Let's assume that $p_{\mathcal{E}} = p_{\mathcal{V}} = p_{\mathcal{I}} = 0$, and there is only one rational miner p_m . In case of an unintentional, or malicious fork, where p_m is one block behind ($z = 1$) and tries to catch-up infinitely long as defined in equation 5.1 from [Nak08, Ros14], his profitability on the fork increases beyond his expected profit for staying on the main chain, when the miner has a hashrate greater than $\approx 38.2\%$ of the total hashrate. This value already occurred in the following publications in the context of cryptocurrencies [TJS16, MHM18], but was not discussed in great detail there, nor related to $1/\phi^2$. We now show how this value is derived and where it comes from.

Proposition 1. When $p_{\mathcal{E}} = p_{\mathcal{V}} = p_{\mathcal{I}} = 0$, then in case of a fork of the only rational miner m with hashrate p_m , the expected normalized reward for the next block of m , who is one block behind ($z = 1$), is higher when trying to catch-up, as defined in equation 5.1 from [Nak08, Ros14], compared to switching to the main chain when $p_m > \frac{1}{\phi^2}$, where ϕ is the golden ratio defined as $\phi = \frac{1+\sqrt{5}}{2} = 1.618033988749895\dots$ s.t. $\frac{1}{\phi^2} = 0.38196601125010515\dots$

Proof. The reason for this lies in the calculation of the success probability for infinitely running attacks as defined in equation 5.1. The parameters for this attack are $p_{\mathcal{B}} = p_{\mathcal{E}} = p_{\mathcal{V}} = p_{\mathcal{I}} = 0$ and p_m , which leads to $p_{\mathcal{A}} = 1 - p_m$. If we insert these parameters into our equations for ρ_{main} and ρ_{fork} we get:

$$\rho_{main} = \frac{\left(1 - \left(\frac{0}{(1-p_m)+p_m}\right)^2\right) \cdot p_m}{(1-p_m) + p_m} = p_m \quad (5.7)$$

This basically describes a linear relationship between hashrate and profit if no attack occurs. In case p_m decides to attack and is one block behind, the profitability translates to equation 5.1 from [Nak08, Ros14]:

$$\rho_{fork} = \frac{\left(\frac{p_m}{1-p_m}\right)^2 \cdot p_m}{p_m} \cdot (0+1) = \left(\frac{p_m}{1-p_m}\right)^2 \quad (5.8)$$

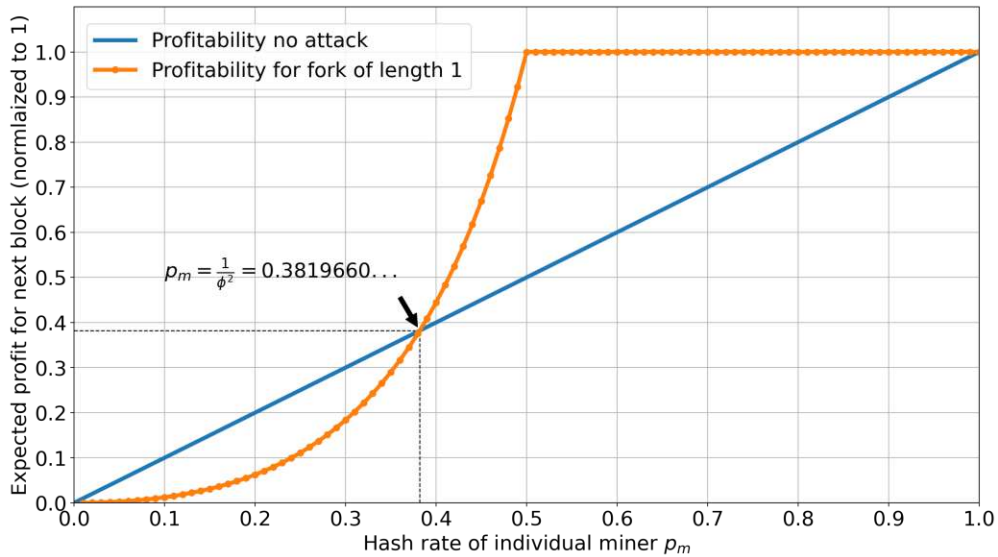


Figure 5.1: The expected reward for the next block ρ is given on the y-axis, while the hashrate p_m of the miner under consideration is given on the x-axis. The figure shows a comparison of the profitability of mining on the main chain without any forks/attack and mining on a fork/attack with $p_B = p_E = p_V = 0$ and $z = 1$.

Finding the intersection (see Figure 5.1) yields.

$$p_m = \left(\frac{p_m}{1 - p_m} \right)^2 = \frac{3}{2} - \frac{\sqrt{5}}{2} = \frac{3 - \sqrt{5}}{2} = \frac{\sqrt{5} - 3}{2} = 1 - \frac{1}{\phi} = \frac{1}{\phi^2} \quad (5.9)$$

Therefore, as soon as the hashrate of the attacking miner p_m surpasses $\frac{1}{\phi^2}$, his chance of winning the infinite race increases beyond his fraction of the total hashrate. \square

Although, theoretically correct, corollary 1 assumes that the attack runs infinitely long, an assumption which is unlikely to hold in practice. Therefore, the $1/\phi^2$ bound can be considered an interesting artefact of the way how these infinite races are modeled

We now will build upon this model from Liao and Katz [LK17] and enhance it by considering the effects of *already contributed blocks* and then consider finite attack durations and economically rational victims.

5.3.2 Considering Already Contributed Blocks

To the best of our knowledge all previous attempts of calculating the attack success probability implicitly assume that all non-attacking miners immediately switch to the

attack chain as soon as it gets ahead of the honest chain. An assumption which is unlikely to hold in practice, as it ignores blocks which have already been appended by those miners to a competing chain. Therefore, if miners should be modeled as economically rational, already mined blocks have to be taken into account. We therefore, enhance the proposed model from [LK17] to account for rational miners and already contributed blocks. As a first step, we extend the model from [LK17] and also consider blocks already contributed by m to the respective chain. For example, if m has already contributed two blocks to the main chain, which would not be rewarded if the attack succeeds, this is denoted by $\eta_{main} = 2$. The number of blocks m has already contributed to the attack chain (if any) is denoted by η_{attack} . It is possible that there are cases where both values ($\eta_{main}, \eta_{attack}$) are greater than 0, so we have to account for that in our calculation. This might happen when m has first worked on the main chain and then switched to an attack. Then, while the attack is still ongoing, m evaluates if it makes sense from her perspective to pursue the progressed attack. The expected reward in number of block rewards (normalized) for one new block in the main chain if the fork/attack fails, is thus given by Equation 5.10. For space reasons we abbreviate $\mathbb{P}(\text{success abstain})$ by $\mathbb{P}(\text{abstain})$.

$$\begin{aligned} \rho_{main\ bl.} := & \\ & \frac{(1 - \mathbb{P}(\text{abstain})) \cdot p_m}{p_{\mathcal{A}} + p_{\mathcal{V}} + p_{\mathcal{I}} + p_m} + (\eta_{main} \cdot (1 - \mathbb{P}(\text{abstain}))) \\ & + (\mathbb{P}(\text{abstain}) \cdot \eta_{fork}) \end{aligned} \quad (5.10)$$

Conversely, the reward in number of block rewards (normalized to 1) for one new block on the attack chain if the attack succeeds is given by Equation 5.11. This assumes that the bribe ϵ is paid for every contributed block on the attack chain, but other variants are also possible. To evaluate an unintentional fork without bribes, ϵ can be set to zero. For space reasons we again abbreviate $\mathbb{P}(\text{success join})$ by $\mathbb{P}(\text{join})$.

$$\begin{aligned} \rho_{fork\ bl.} := & \\ & \left(\frac{\mathbb{P}(\text{join}) \cdot p_m}{p_{\mathcal{B}} + p_{\mathcal{E}} + p_m} \cdot (\epsilon + 1) \right) \\ & + (\eta_{fork} \cdot (\epsilon + 1) \cdot \mathbb{P}(\text{join})) + (\eta_{main} \cdot (1 - \mathbb{P}(\text{join}))) \end{aligned} \quad (5.11)$$

Given these Equations, a rational miner now can compare the achievable rewards on both chains and pick the most profitable one. In this way, also the case in which a miner has contributed blocks to two or more chains can be captured and compared. Figure 5.2 shows a comparison for different hashrates and already contributed blocks. It can be observed that, as soon as a miner has already contributed a block to a chain, his expected profit increases significantly on that chain. This makes it unlikely, that rational players in such a situation will readily switch to the attack chain as soon as it takes the lead. Moreover, it can be observed that, if there is another attacker ($p_{\mathcal{B}} > 0$), joining an attack becomes more profitable than staying on the main chain sooner, i.e., with lower hashrate p_m . Although, staying on the main chain is slightly more profitable if somebody else

5. HOW MUCH IS THE FORK?

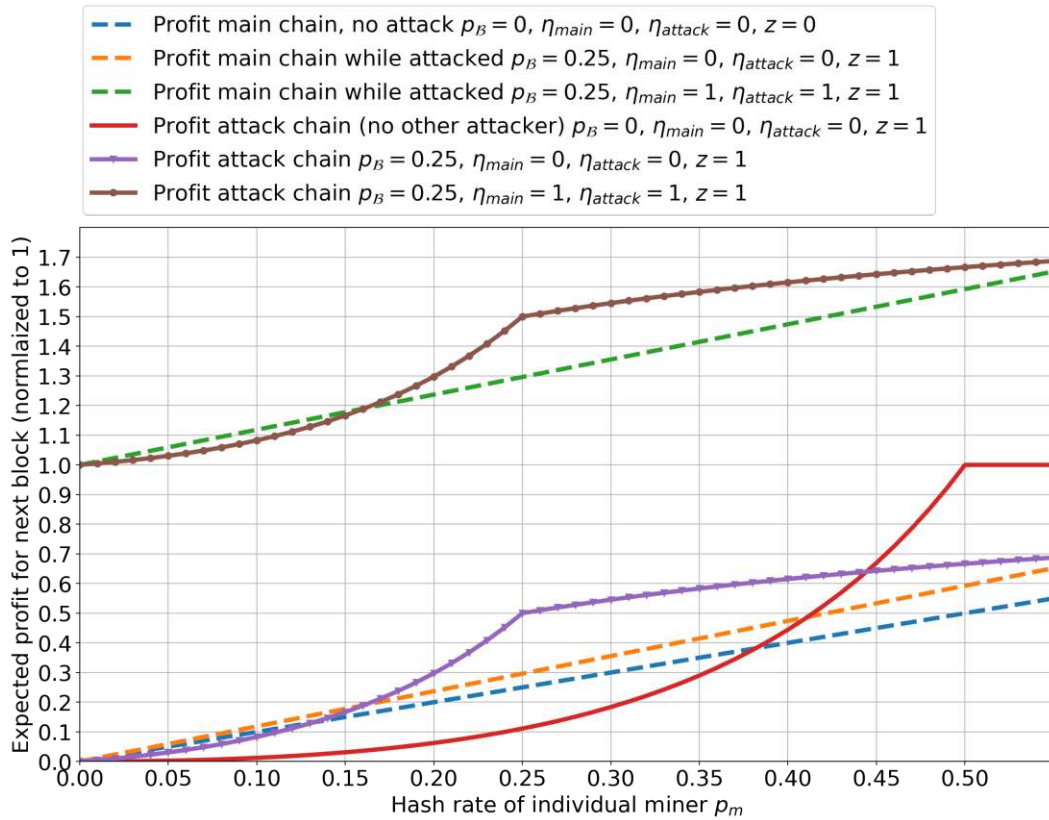


Figure 5.2: The expected reward (ρ) for the next block is given on the y-axis, while the hashrate p_m of the miner under consideration is given on the x-axis. The figure shows a comparison of the profitability of different scenarios for mining on the main chain, versus mining on the attack chain.

with low chances of success is working on an attack chain, as in this case less hashrate is concentrated on the main chain. The same holds true, in the other direction, but since the overall hashrate on the attack chain is lower, the potential gains are higher as they have to be divided amongst fewer players.

As the probability calculation in this case implicitly still assumes that victims will switch immediately as soon as their chain falls behind and that attackers will stick to their chain infinitely long, we now augment the model to account for rational victims and finite attacks.

5.3.3 Considering Economically Rational Victims and Finite Attacks

We now want to model rational miners, who do not immediately switch to the attack chain once it has taken the lead, e.g., because they have already contributed blocks to the main chain which they would lose in this case. The miners who keep on mining on the main chain can be viewed as *victims* (\mathcal{V}) with hashrate $p_{\mathcal{V}}$, whereas miners who switch to the attack chain as soon as it takes the lead can either be viewed as *altruistic* (\mathcal{A}) with hashrate $p_{\mathcal{A}}$, or as *indifferent* (\mathcal{I}) with hashrate $p_{\mathcal{I}}$. Miners who start working on a fork/attack chain can be viewed as *Byzantine* (\mathcal{B}) with hashrate $p_{\mathcal{B}}$, or if they are economically rational as *extractor* (\mathcal{E}) with hashrate $p_{\mathcal{E}}$. Those last two sets of miners would profit if the attack chain wins.

We model an attack, which is not immediately over as soon as the attack chain takes the lead, using a finite Markov chain. Since we are only interested in modeling practical and thus finite attacks, it is sufficient to use a finite Markov chain for the practically plausible range in which both chains will grow relative to each other.

Thereby, the set of victims (\mathcal{V}) will work against the attack, even if their (previous main) chain is behind already. They will only give up if a configurable lead (\vec{k}), in terms of blocks, is reached. In opposition to that, the attacker and bribable rational miners work on the attack chain ($\mathcal{B} \cup \mathcal{E}$) of the fork. The third fraction of miners consisting of altruistic, as well as indifferent rational miners, work either for, or against the attack, depending on which chain is currently the longest ($\mathcal{A} \cup \mathcal{I}$). The difference between the last two groups is, that altruistic miners will never support a shorter chain, while the decision of indifferent miners can be subject to change depending on their expected profit⁴.

As we are only interested in practical attacks, they have a certain finite maximum duration. This is the number of blocks (N) the fork is assumed to last, i.e., the period it can be financed by the involved parties. In our model this is the number of steps that are taken in the Markov chain (see Figure 5.3). Moreover, each of the two opposing parties (\mathcal{V} against $\mathcal{B} \cup \mathcal{E}$) has a certain pain threshold in terms of blocks that their chain can fall behind until they deem it unlikely that they will ever catch up again. For \mathcal{V} this number in terms of blocks is defined by \vec{k} , whereas for $\mathcal{B} \cup \mathcal{E}$ this value is \overleftarrow{k} . The winning condition, from the perspective of the attackers, for this kind of race can be defined in two ways:

- **Win:** The sum of all probabilities of the states on the right hand side, where the attack chain is the longest chain after a total of N blocks i.e., steps taken in the Markov chain.
- **Clear win:** The success probability of the *clear win* state at which the attack chain has an advantage of \vec{k} s.t., the victims will give up.

⁴For the analysis presented here, these two groups (\mathcal{A} and \mathcal{I}) are treated similarly. If this should not be the case, the Markov chain can be augmented. For example the hashrate $p_{\mathcal{I}}$ can be modeled to work for, or against the attack, if a chain is in the lead for a sufficiently large number of blocks.

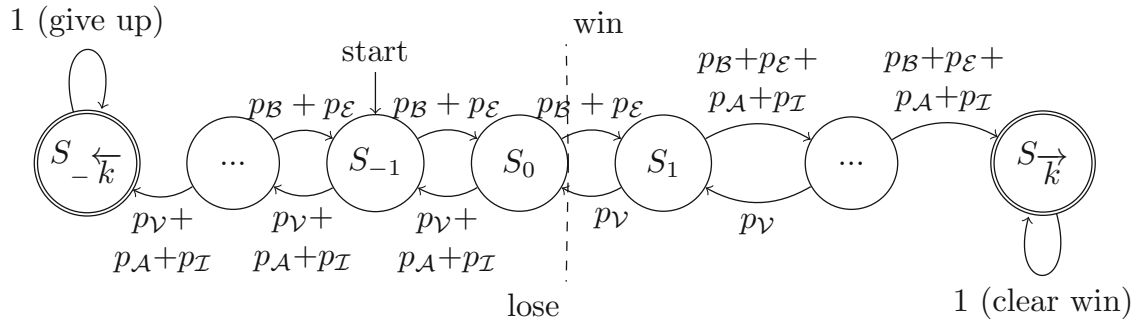


Figure 5.3: Markov chain for modeling finite attacks and persistent victims (p_v).

Every other state is a lose state for the attackers. The start state is defined by the initial disadvantage for the attackers z . Usually for block exclusion (also known as censorship) attacks, and missed MEV opportunities, $z = 1$, thus S_{-1} is the start state.

To validate our approach, we now want to use our Markov chain model to approximate the success probability of a classical infinite attack⁵, in which the defenders are not rational and give up as soon as they are behind. Therefore, we have to configure our Markov chain model as follows: Set $\overrightarrow{k} = 1$ and increase N as well as \overleftarrow{k} to approach the success probability of the infinitely running attack. Figure 5.4 shows that our Markov model approaches the maximum success probability of an infinitely running attack as N grows.

We now use the probability of all success states after N iterations of our Markov chain, to replace the probability calculation in Equation 5.10 and 5.11 with the success probability of the attack computed via the Markov chain, instead of using the infinite success probability calculation. Thereby, we can also compute the profitability using the same formula, but now with a different probability calculation. Before we compare some example scenarios using this new model, we extend it to also account for attacks in which a briber compensates participating bribees for contributed blocks, even if the attack as a whole is not successful.

5.3.4 Effort-Related Compensation for Contributed Blocks

Previous works indicate that paying bribes even if the attack as a whole is not successful is a viable strategy. In paper [WHF19] the authors describe effective transaction censorship attacks which operate by paying for complacent blocks. In paper [JSZ⁺19] a double-spend attack is described in which an attacker compensates already contributed blocks to the attack branch, even if the attack as a whole is not successful. To evaluate the costs of the

⁵Where q is the attacker hashrate s.t. $\left(\frac{q}{1-q}\right)^{z+1}$, as defined in [Nak08, Ros14]

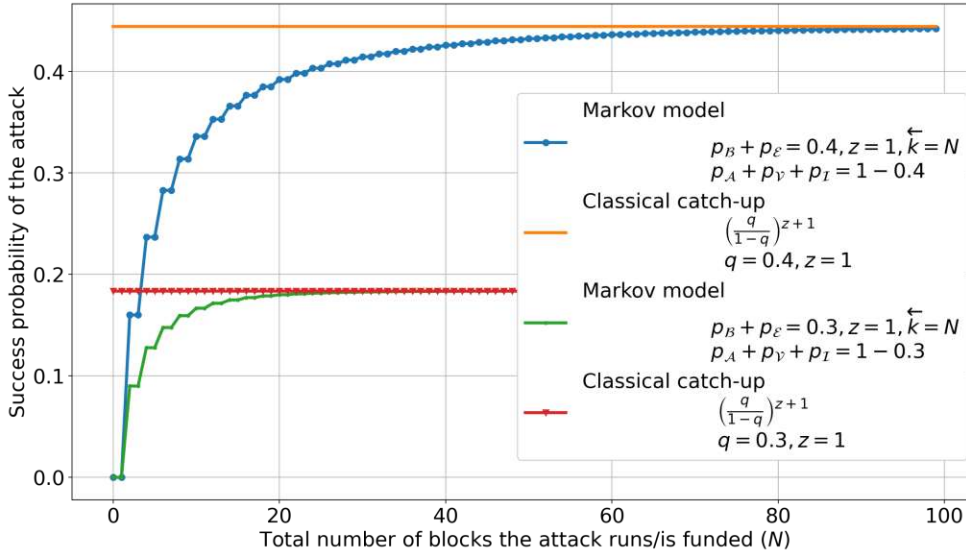


Figure 5.4: Comparison of the infinite attack success probability to our finite Markov chain model for increasing attack duration $N = \overleftarrow{k}$ (x-axis).

overall attack the authors in paper [JSZ⁺19] simulated different scenarios and thereby focused on the perspective of the briber.

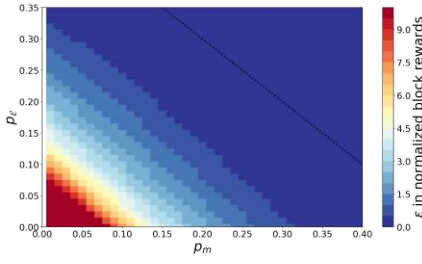
In this chapter we want to illustrate the economic feasibility in such a scenario from the perspective of an individual bribee, i.e., an economically rational miner m . Therefore, we augment the expected profitability calculation for the next block on either chain in a way that ensures that already contributed blocks are compensated even if the attack is not successful.

To calculate the expected profit for the next block on the main chain in this case, Equation 5.10 is extended to also include compensations for blocks contributed to the fork. At first this might seem counter intuitive, but this captures the case where a miner m has already contributed blocks to a fork and considers switching back to the main chain. In this case m would unconditionally receive the compensation for already contributed attack blocks even if the main chain wins, plus a bribe for the attack chain blocks if the fork happens to be successful and the main chain loses.

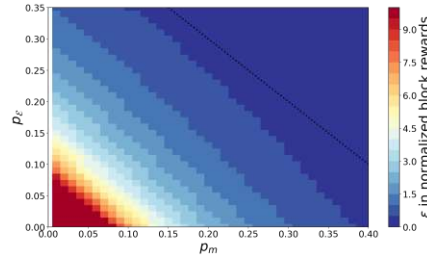
$$\rho_{main\ comp.} := \frac{(1 - \mathbb{P}(\text{abstain})) \cdot p_m}{p_A + p_V + p_I + p_m} + (\eta_{main} \cdot (1 - \mathbb{P}(\text{abstain}))) \quad (5.12)$$

$$+ \eta_{fork} + (\eta_{fork} \cdot \epsilon \cdot \mathbb{P}(\text{abstain})) \quad (5.13)$$

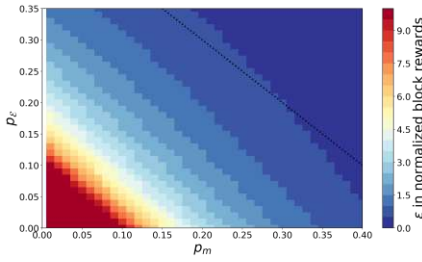
5. HOW MUCH IS THE FORK?



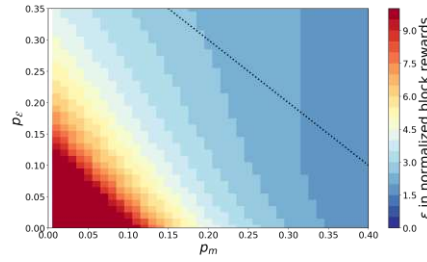
(a) *Model*: Min. ϵ for $\rho_{fork} > \rho_{main}$, with classic infinite probability calculation (Equation 5.6). *Conf.*: $z = 1, N = \infty$



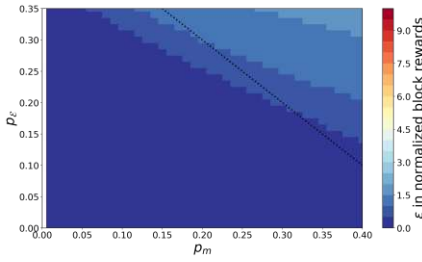
(b) *Model*: Min. ϵ for $\rho_{fork} > \rho_{main}$, with probabilities from Markov chain *Conf.*: $p_V = 0, z = 1, \vec{k} = 1, \overleftarrow{k} = 6, N = 6$



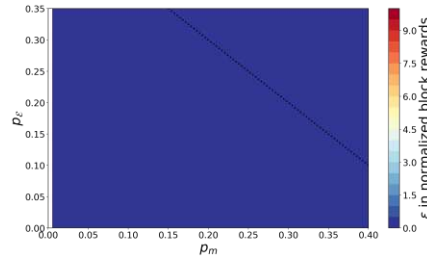
(c) *Model*: Min. ϵ for $\rho_{fork} > \rho_{main}$, with probabilities from Markov chain *Conf.*: $p_V = 0.25, z = 1, \vec{k} = 3, \overleftarrow{k} = 6, N = 6$



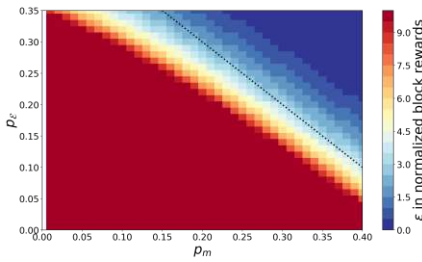
(d) *Model*: Min. ϵ for $\rho_{fork bl.} > \rho_{main bl.}$, with probabilities from Markov chain *Conf.*: $p_V = 0.25, z = 1, \vec{k} = 3, \overleftarrow{k} = 6, N = 6, \eta_{main} = 1$



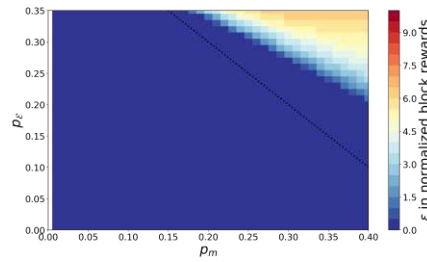
(e) *Model*: Min. ϵ for $\rho_{fork comp.} > \rho_{main comp.}$, with probabilities from Markov chain *Conf.*: $p_V = 0.25, z = 1, \vec{k} = 3, \overleftarrow{k} = 6, N = 6, \eta_{main} = 1$



(f) *Model*: Min. ϵ for $\rho_{fork comp.} > \rho_{main comp.}$, with probabilities from Markov chain *Conf.*: $p_V = 0.25, z = 1, \vec{k} = 3, \overleftarrow{k} = 6, N = 6$



(g) *Model*: Min. ϵ for $\rho_{fork bl.} > \rho_{main bl.}$, with probabilities from Markov chain *Conf.*: $p_V = 0.25, z = 6, \vec{k} = 3, \overleftarrow{k} = 9, N = 10, \eta_{attack} = 3$



(h) *Model*: Min. ϵ for $\rho_{fork comp.} > \rho_{main comp.}$, with probabilities from Markov chain *Conf.*: $p_V = 0.25, z = 6, \vec{k} = 3, \overleftarrow{k} = 9, N = 10, \eta_{main} = 3$

Figure 5.5: Minimum bribe value per block ϵ , given in normalized block rewards, for different models and configurations. Thereby, ϵ reaches from 0 (blue) to 10 or higher (red). The y-axis shows p_E , while the x-axis shows p_m . The dashed line marks $p_E + p_m = 0.5$.

To calculate the expected profit for the next block on the attack chain in this case, Equation 5.11 is extended to also include compensations for blocks contributed to the fork. In this case, there additionally exists the probability to also get a normalized block reward (excluding bribe ϵ) for the next block if the attack is not successful. Additionally, all already contributed blocks are compensated even if the attack fails (without additional ϵ per block). In case the attack is successful, an additional bribe ϵ is paid per block.

$$\begin{aligned} \rho_{fork\ comp.} := & \\ & \frac{\mathbb{P}(\text{join}) \cdot p_m}{p_B + p_E + p_m} \cdot (1 + \epsilon) + \frac{(1 - \mathbb{P}(\text{join})) \cdot p_m}{p_B + p_E + p_m} \\ & + \eta_{fork} + (\eta_{fork} \cdot \epsilon \cdot \mathbb{P}(\text{join})) + (\eta_{main} \cdot (1 - \mathbb{P}(\text{join}))) \end{aligned} \quad (5.14)$$

5.3.5 Comparison

We now want to compare different attack scenarios regarding the required minimum bribe ϵ for an attack. To calculate the min. bribe s.t. switching to the attack chain becomes more profitable for m compared to mining on the main chain, we again set $\rho_{fork^*} > \rho_{main^*}$ and solve for ϵ .

Figure 5.5 shows color map plots of the minimum bribe ϵ required such that contributing the next block to the attack chain is more profitable for a miner with hashrate p_m (x-axis) than extending the main chain. The y-axis shows the hashrate (p_E) of other economically rational attackers. The dashed black line marks $p_m + p_E > 0.5$, at which point the attackers would be able to overtake the system when attacking infinitely long. The required minimum bribe ϵ is denominated in normalized block rewards in a range from 0 to 10, where 0 is blue, and a bribe of 10 (or more) times the normalized block reward is red.

Figure 5.5a to Figure 5.5f show various models and configurations for the case $z = 1$. This resembles the case of a one-block fork as it might happen when certain transactions ought to be censored, or a missed MEV opportunity should be exploited although a block has already been mined. Figure 5.5a depicts the case where no compensations for already contributed blocks to the attack chain are paid (Equations 5.10 and 5.11 are used in the inequality). Furthermore, in Figure 5.5a the probability is calculated using the classic infinite model from Equation 5.1. In comparison, Figure 5.5b uses our Markov chain with an attack duration of $N = 6$ and $\vec{k} = 1$, which resembles the case that the attack is over as soon as the attack chain takes the lead. In both figures no compensations for already contributed blocks are paid. It can be observed that in this comparable scenario, the required bribes are slightly higher in the Markov chain model, as in this case the attack does not run infinitely long (although the infinite case can be approximated by increasing N).

Figure 5.5c shows the same situation in our Markov chain model, but now the victim has hashrate $p_V = 0.25$. It can be seen that in this case the required bribes are of course higher as now the victim is working against the attack even after the attack chain has

become the longest chain. The bribe further increases in Figure 5.5d as now m has contributed already one block to the main chain.

Figure 5.5e suddenly does require hardly any bribes as now the Equations 5.12 and 5.14 for effort-related compensation are used. In this case, even though m has already contributed one block to the main chain ($\eta_{main} = 1$), no extra direct bribe per block (except for the compensations) is required. This effect is further amplified in Figure 5.5f, where effort-related compensation is used as well, but no blocks have been contributed by m to any chain in this case. Here we see that no extra bribes per block are required such that extending the attack chain is more profitable for m than extending the main chain. Therefore, it would make sense to combine the exploitation of a missed MEV opportunity, with a bribing attack that compensates already mined blocks, as hardly any additional bribe per block is needed to incentivize economically rational miners to participate.

Figure 5.5g shows a classical double-spend scenario where the attack chain is $z = 6$ blocks behind. In this case the required bribe is very high although m has already contributed $\eta_{attack} = 3$ blocks to the fork. This is because in this case no effort-related compensation is paid for already contributed blocks. As soon as effort-related compensation is paid for contributed blocks the required bribe again becomes 0 even for longer range forks like that. Figure 5.5h shows such a case even in a scenario where m has already contributed blocks to the main chain ($\eta_{main} = 3$). Even in this case no extra bribe is required to incentivize m to mine on the attack chain if effort-related compensation is used. Observe that in Figure 5.5h, bribes are required as soon as the hashrate of $p_m + p_{\mathcal{E}}$ is high enough. The reason for this is that the probability to find a block on the main chain gets better compared to the attack chain. Moreover, the stakes are higher on the main chain as 3 blocks have already been contributed which would be lost in case the main chain loses.

5.4 Implications of our Simulations

In this chapter, we have presented an improved model for calculating the probability and profitability of chain forks, i.e., attacks, from the perspective of an individual miner m . Our model considers, configurable finite attack durations, already contributed blocks to the respective chains, as well as victims which do not immediately switch to the attack chain as soon as it takes the lead. We have applied the model to more accurately investigate forks of length 1, which are the relevant case for exploiting missed MEV opportunities. Furthermore we have described approaches which allow modeling bribing attacks which compensate miners for already contributed blocks, even if the overall attack is unsuccessful. We have shown that bribing attacks, which can plausibly ensure that blocks contributed by miners are compensated, require hardly any additional bribe per block to incentivize economically rational miners to participate in the attack. This further emphasizes the risk such bribing attacks pose to the overall stability of the underlying system, especially for short range forks.

In Chapter 6 we investigate the long term consequences of successful attacks. For example, a drop in the exchange rate which reduces the profitability of the attack. Such

5. HOW MUCH IS THE FORK?

considerations have to be taken into account by economically rational miners as well. If the exchange rate is not static, then also the amount of funds players are currently holding becomes relevant.

Estimating (M)EV and its Consequences

In this chapter¹, the problem of estimating the miner extractable value of other participants in an NC based cryptocurrency is illustrated and described in relation to algorithmic incentive manipulation. The term *miner extractable value* (MEV) has been coined to describe the value which can be extracted by a miner, e.g., from manipulating the order of transactions within a given timeframe. MEV has been deemed an important factor to assess the overall economic stability of a cryptocurrency. This stability also influences the economically rational choice of the security parameter k , by which a merchant defines the number of required confirmation blocks in cryptocurrencies based on Nakamoto consensus. Unfortunately, although being actively discussed within the cryptocurrency community, no exact definition of MEV was given when the term was originally introduced. In this chapter, we outline the difficulties in defining different forms of extractable value, informally used throughout the community. We show that there is no globally unique MEV/EV which can readily be determined, and that a narrow definition of MEV fails to capture the extractable value of other actors like *users*, or the probabilistic nature of permissionless cryptocurrencies. We describe an approach to estimate the *minimum* extractable value that would incentivize actors to act maliciously and thus can potentially lead to consensus instability. We further highlight why it is hard, or even impossible, to precisely determine the extractable value of other participants, considering the uncertainties in real-world systems. Finally, we outline a peculiar yet straightforward technique for choosing the individual security parameter k , which can act as a workaround to transfer the risk of an insufficiently chosen k to another merchant.

¹This chapter represents an extended version of publication [JSSW22a].

6.1 The History of (Miner) Extractable Value

The term *miner extractable value* was first introduced by Daian et al. [DGK⁺20] to refer to the value which can be extracted by a miner from manipulating the order of transactions within the blocks the respective miner creates. This ability can be used for *front-running* attacks [EMC19], which can lead to guaranteed profits through token arbitrage, or other related types of attacks like *back-running* and combinations thereof [QZG22]. Such attacks, which exploit the ability of miners to arbitrarily order transactions within their blocks, are feasible and currently observed in practice [ZQC⁺21, ZQT⁺20, TIGS21, ZQG21]. Thus, order fairness evidently poses an issue for prevalent permissionless PoW cryptocurrencies [KZGJ20]. When the stakes are sufficiently high, MEV can even incentivize blockchain forks and thereby also have consequences not only on transactions in future blocks, but also for the underlying *consensus-layer security*, as also noted by Daian et al. [DGK⁺20] and Zhou et al. [ZQC⁺21]. Related attacks aimed in this direction are *undercutting attacks* [CKWN16], *time-bandit attacks* [DGK⁺20], or more broadly: Attacks involving economic incentives in general, such as any form of *bribing attack* [Bon16, MHM18, Bon18, LK17], which have been summarized under the term *algorithmic incentive manipulation* [JSZ⁺21b].

Given all these attacks and their far-reaching consequences, MEV undoubtedly is an essential concept when reasoning about economic stability aspects and cryptocurrency security under economical considerations. Recently, a related term called *blockchain extractable value* [QZG22] (BEV) was introduced. BEV refers to the value extractable by different forms of front-running attacks which are not necessarily performed by miners, but *users*. Unfortunately, in the case of MEV as well as in the case of BEV, no exact definition was given by the authors when the term was originally introduced ².

As the concept of MEV/BEV is tied to the economic incentives of whether or not to fork a certain block/chain [ZQC⁺21], this question also relates to economic considerations regarding the choice of the personal security parameter k of merchants. An accurate estimation of the overall MEV/BEV value, would allow to adapt and increase k accordingly in periods of high overall MEV/BEV. The choice of the security parameter k , which determines the number of required confirmation blocks until a payment can safely be considered confirmed, has been studied in a variety of works. Rosenfeld [Ros14] showed that, although waiting for more confirmations exponentially decreases the probability of successful attacks, no amount of confirmations will reduce the success rate of attacks to 0 in the probabilistic security model of PoW, and that there is nothing special about the often-cited figure of $k = 6$ confirmations. Sompolinsky and Zohar [SZ16] defined different acceptance policies with different error probabilities and use cases. According to [SZ16] an acceptance policy that is resilient to a double-spend anywhere in the chain cannot rely on a static parameter k , but has to be logarithmic in the chain's current length. Garay, Kiayias and Leonardos [GKL20] defined the security parameter k , after

²Concurrent work [BDKJ21, OSS⁺21], also attempts to initially define different forms of *extractable value* and is compared to this chapter in Appendix A.8.1 and A.8.2.

which a transaction can be considered part of the *common prefix*, as a function of the security parameter of the hash function (κ), the typical number of consecutive rounds for which a statement would hold, and the probability of at least one honest party finding a valid PoW in a round. In spite of these well-founded theoretical results under honest majority assumptions, in practice there is no global and agreed-upon security parameter k for prevalent PoW cryptocurrencies. Instead merchants choose their individual k on a best-practice basis, taking their individual economical risk into account. Since the likelihood of potential forks increases when sufficiently large extraction opportunities for miners arise, the question on how to define and estimate the extractable value of a block (or a chain) also relates to the choice of the personal security parameter k . This has also been studied in the context of bribing, especially in the *whale attack* [LK17], where large fees are offered on competing chains, as well as in context of incentives for miners to fork high value blocks when the majority for the block reward comes in form of transaction fees [CKWN16, TE18].

Contribution: In this chapter we *describe different forms of extractable value* and how they relate to each other. Furthermore, we outline a series of observations which highlight the *difficulties in defining these different forms of extractable value* and why a generic and thorough definition of it (and thus its precise calculation) is *impossible for permissionless cryptocurrencies* without assuming bounds regarding all available resources (e.g., other cryptocurrencies) that are, or could be of relevance for economically rational players. We also describe a way to estimate the minimum extractable value, measured in multiples of normalized block rewards of a reference resource, to incentivize adversarial behavior of participants which can lead to consensus instability³. In the end, we propose a peculiar yet straightforward technique for choosing the personal security parameters k regardless of extractable value opportunities.

6.2 Economic Rationality and Extractable Value

Rationality depends on the criteria which should be optimized. This could be reward in terms of cryptocurrency units in some PoW cryptocurrency, or a more abstract criteria such as the overall robustness of the cryptocurrency ecosystem. We start out with a simplified definition of *economic rationality* to model the preferences of *actors*, or *parties*⁴ within the system. Hereby, actors are divided into two disjoint sets: *miners* (\mathcal{M}) and *users* (\mathcal{U}), where $\mathcal{M} \cup \mathcal{U} = \mathcal{P}$. Compared to miners, users do not having any direct voting power in the system (e.g., hashrate). Whenever, we refer to rational within this work, we refer to the definition of *economically rational in \mathcal{R}* :

Definition 2 (Economically rational in \mathcal{R}). An actor (i) is **economically rational**, with respect to a finite non-empty set of resources $\mathcal{R}_i := \{R_0, R_1, \dots\}$, when it is his

³All source code and artefacts can be found on GitHub https://github.com/kernoelpanic/estimatingMEVishard_artefacts

⁴Also the term *players* is commonly used to refer to the involved parties.

single aim to maximize his profits measured in these resources. To also map the individual preferences of an actor regarding this set of resources, the quantities of all resources from that actor are converted into **value units**. Therefore, from the perspective of an actor i each resource R is a tuple $\langle r, e \rangle$ consisting of: The quantity r the actor i holds in the respective resource; and the individual exchange rate e for the conversion in value units which reflect the individual preference of that actor.

A quantity of a certain resource $\langle r, e \rangle$, which is optimized by a rational actor (indexed by i), is denoted as $f_i(r, e)$. This function returns the *value units*, also referred to as *funds*, actor i has in r , calculated using his individual exchange rate e for r . If the exchange rates are the same for every party, or clear from the context, they can also be omitted. For practical purposes and to aid comparability, we will use *normalized block rewards* of a reference resource (e.g., Bitcoin block rewards including average fees) as *value unit* in which all funds are denoted. Therefore, all exchange rates of other resources convert their respective quantities into normalized block rewards of the reference resource.

In other words, value units can also be thought of as fiat currency, which in turn can be received in exchange for cryptocurrency units. In this chapter, value units are measured in multiples of the block reward of a reference cryptocurrency, which is the first resource in \mathcal{R}_i (i.e., R_0). A resource can be anything of value to an actor e.g., a cryptocurrency, a token within a cryptocurrency, or a fiat currency. If not stated differently, all parties care about the same set of resources and the exchange rate for each resource is globally defined (e.g., by an exchange service) and thus the same for every actor. This means, for the simplest case where all parties only care about one resource and have the same global exchange rate (e_{global}), we have: $\forall i \in \mathcal{P} (|\mathcal{R}_i| = 1 \wedge e \in R_0 \wedge e = e_{global})$, where the valuation of an actor i is given by $f_i(r, e_{global})$, or abbreviated just $f_i(r)$. We start our evaluation with such a scenario.

Summing up: By our definition of economic rationality actors want to maximize their overall funds (valuation) from all their resources, which are measured in value units, i.e., $f_i(\mathcal{R}_i) = \sum_{\langle r_j, e_j \rangle \in \mathcal{R}} f_i(r_j, e_j)$.

6.2.1 Miner Extractable Value

To calculate the gain or profit an actor has made within a chain of blocks c , with their sequence of transactions τ , it is essential to estimate the costs as well as the extractable value for the respective actor. This has been done in previous works [DGK⁺20, QZG22] mostly by analyzing past Ethereum blocks with their associated transactions, while looking for profitable trades on the blockchain in retrospect. The gathered data was then also used when analyzing how to automatically detect and exploit such trading situations [ZQC⁺21, ZQT⁺20]. In this context, the term *miner extractable value* (first introduced by Daian et al. [DGK⁺20]) was informally used to describe the value which can be extracted by a miner by including a certain transaction in terms of fees, or guaranteed profits through token arbitrage. We now provide a definition within the context of our model and in accordance to the literature. Therefore, we first focus on a

scenario where we assume that there is only one resource, as well as one exchange rate, which is the same for every actor.

Definition 3 (Miner Extractable Value $MEV_i(c)$). The miner-extractable value $MEV_i(c)$, describes the total value (denominated in value units), transferred to a **miner** i from a sequence of transactions τ , included and thus mined in the respective chain of blocks c , which is part of the main chain.

This definition focuses on identifying the extractable value in retrospect and is thus suited for empirically analyzing the MEV of historic blocks. We extend this to a more forward looking definition later in Section 6.2.3, where we also take the probabilistic nature of most cryptocurrencies into account.

Regardless of the time when the MEV is determined, there are couple of characteristics all types of MEV have in common. First, the total extractable value depends on the type of optimization the miner i is performing. If transactions are only ordered by fee, the miner extractable value can be expressed as the “usual” mining reward from fees and block rewards i.e., $MEV_i(c) := FEE(c) + BLOCKREWARD(c)$. If there is a value gain from received transactions, or rewards from performed token arbitrage and order optimizations, then the respective income opportunities, e.g., income from received transactions and attacks, have to be taken into account as well. In other words, a general definition of MEV describes the value (i.e., the reward) which can be extracted by a miner, extended by additional revenue opportunities originating from the capabilities to interact with the system the miner is tasked to validate. This leads to the question: What *possibilities* to extract value are available to miners?

This question already outlines why the concrete amount of MEV is difficult to generalize for all miners, as it can be different for every individual miner. The MEV of a given miner i depends on the type of value extraction optimization the respective miner is capable and willing to perform. Hereby, the possibilities reach from simple fee optimization techniques, like selecting the transactions which provide the highest fees, over order optimization (for example, attempting to maximize gas consumption in smart contracts⁵), to participating in sophisticated front-running attacks. Moreover the MEV is affected by the information available to the miner (e.g., her view of the transaction pool). Therefore, we can make the following observation:

Observation 1. The miner extractable value (MEV) is different for every miner depending on the optimization techniques the miner is capable and willing to perform, as well as other transactions which directly affect her individual revenue during the respective period (e.g., received funds).

⁵Note that, a naive algorithm for finding the optimal ordering of all transactions is factorial in the number of transactions, which is computationally infeasible even for most current Ethereum blocks which have more than 200 transactions.

In other words, there may exist a miner m_1 that has a higher MEV than a miner m_2 because he has received a large incoming payment transaction in the same sequence of transactions $\tau \in c$, i.e., $\text{MEV}_{m_1}(c) > \text{MEV}_{m_2}(c)$. Therefore, miner m_2 may be more willing to participate in an attack which changes the past blocks c than the miner m_1 .

6.2.2 Extractable Value

Since not all attacks (or more generally, ways to maximize profit) necessarily require the capabilities of a miner, the given definition of MEV does not capture these. front-running attacks for example, can be performed by actors which do not necessarily have to be miners themselves, but can be *users* instead. From their perspective the definition of MEV does not apply, as they are not capable of mining a block on their own (as they have zero hashrate).

Observation 2. The definition of MEV is focused on *miners* and does not capture opportunities for *users* to gain (more) value units from a certain sequence of transactions τ than from another sequence of transactions τ' .

In contrast to MEV, *blockchain extractable value* (BEV) [QZG22], was previously described in a broader context and thus also refers to the value extractable by different forms of front-running attacks which are not necessarily performed by miners. As recent analysis [DGK⁺20, QZG22, ZQC⁺21, ZQT⁺20, TIGS21] show, front-running is performed by bots, which bid for an early slot in a block by raising the miner extractable transaction fee⁶. However, these earlier discussions regarding BEV [QZG22] omit a precise definition, which we provide in the general form of *extractable value* (EV_i), for the amount that is transferred to any actor i from a given blockchain c , or sequence of transactions τ .

Definition 4 (Extractable Value $\text{EV}_i(\cdot)$). The extractable value $\text{EV}_i(\cdot)$, describes the total incoming value, subtracted by the outgoing value, which is transferred to actor i from a transaction, or sequence of transactions τ , if it is included and thus mined in the respective chain of blocks c , which is part of the main chain.

Using this definition of extractable value, the miner extractable value can also be defined as $\text{EV}_i(c)$ of any miner $i \in \mathcal{M}$. As with miner extractable value, the extractable value of different actors i and j for the same chain of blocks c can also be different. Again $\text{EV}_i(c)$ and $\text{EV}_j(c)$ depend on whether they have received, or sent, transactions within this chain or not.

In other words, observation 2 shows that MEV is just a way to extract value which can be executed by a subset of actors i.e., miners. Since we also know from observation 1 that there is no globally unique MEV, which is the same for all miners, the same argument

⁶In Ethereum the extractable fee is a combination of `gasPrice` multiplied by `gasUsed`.

can also be extended to EV. So there also cannot be a globally unique EV, that is the same for all actors. The question is if the EV can be meaningfully estimated, or bounded? Assume we have two parties i and j , where j wants to estimate the extractable value of i for a certain chain c . If j wants to estimate $EV_i(c)$, then this means j has to attribute all transactions generating value for i in c correctly. If the respective actor i pseudonymously performs and receives transactions under various different addresses (or in other privacy preserving ways like shielded transactions in Zcash [Zca]), this hampers the correct estimation of $EV_i(c)$ for any third party j which does not know which transactions belong to a certain actor.

Observation 3. If there are transactions in c that are not uniquely attributable to other parties from the perspective of actor j , then the upper bound of the $EV_i(c)$ for a certain actor i is the total value transferred by those non-attributable transactions.

This means, for known finite chains of blocks in certain cryptocurrencies where the overall value that has been transferred is observable, the extractable value can be upper-bounded in retrospect as soon as the respective chain is known.

Note that, even if all transactions can be correctly attributed to an actor, the exact effect of an outgoing transaction on the EV in a smart contract capable cryptocurrency is still difficult to measure. Generally, all outgoing transactions reduce the EV, as the miner loses funds. Although, if the outgoing transaction is a guaranteed profit token arbitrage, or other profitable type of front-running, the EV might very well increase. We omit the details of analyzing the EV of a particular transaction in a smart contract capable cryptocurrency and refer to a strain of research dealing with this topic [DGK⁺20, ZQC⁺21, ZQT⁺20, TIGS21, ZQG21].

6.2.3 Expected Extractable Value

So far we have only considered the extractable value of past blocks in retrospect ($MEV_i(c)$), or blocks and transactions under the assumption that they eventually will make it into the main chain ($EV_i(c)$, or $EV_i(\tau)$). Hereby, we did not account for the probability with which the estimated value can be realized. As mining in prevalent cryptocurrencies is a stochastic process, getting a certain chain accepted into the common-prefix depends on several factors - one of which being the hashrate that supports a given chain⁷. Therefore, it is more appropriate to refer to the *expected extractable value* (EEV) when comparing potential/future rewards of mining strategies, pending blocks, or forks.

Definition 5 (Expected Extractable Value $EEV_i(\cdot)$). The expected extractable value $EEV_i(\cdot)$, describes the total incoming value in value units, subtracted by the outgoing value, which is transferred to actor i on expectation using a certain strategy which produces a transaction, sequence of transactions (τ), or blocks (c) that later become part of the main chain with some probability.

⁷Another one being propagation times, but we will ignore that for now.

To maximize the EEV, actors will pick an according strategy. Hereby, the probability space depends on the concrete model under which a given strategy is analyzed.

Observation 4. In permissionless PoW cryptocurrencies, the number of participants $|\mathcal{P}|$ is theoretically unbounded, as miners can join and leave at any time. Thus the sample space of possible events is theoretically unbounded as well.

Therefore, certain assumptions e.g., regarding the available resources of (potential) players are required, to bound their influence (i.e., action space) on a given cryptocurrency.

We now describe a simple idealized system model and strategy and use it as a reference to compare deviating strategies and changes in the model against this setting. We therefore, assume that there is only one resource of interest and no further actors join the cryptocurrency.

Definition 6 (The strategy HONEST). We define the strategy HONEST for miners in a cryptocurrency R , as the process of always extending the currently known longest (heaviest) chain and immediately publishing and forwarding every found block and transaction.

If every miner plays HONEST and has a constant hashrate, then this results in an infinitely repeated game, in which every miner receives exactly the reward that is proportional to his hashrate. Therefore, HONEST satisfies *ideal chain quality* [GKL14], as the percentage of blocks in the blockchain of every actor is exactly proportional to their individual hashing power⁸. We assume that this is the desired ideal state in which the system should be and thus the goal of the mechanism design. Moreover, it is more-or-less the empirically observed behavior of miners as serious deviations are rarely observed in mainstream cryptocurrencies⁹.

Assume a miner i that does not receive or send any transactions (apart from collecting rewards from mined blocks), and that the extractable value is given in normalized block rewards (including fees) as a value unit. Then if everybody plays HONEST, the strategy HONEST for a sequence of n blocks, would have the EEV depicted in equation 6.1. The strategy would be profitable if the mining costs ($costs_{mining}$) for mining the respective number of blocks is lower than the EEV. This also assumes that the hashrate (p_i) of actor i is common knowledge and static for the duration of the evaluation.

⁸Note that, in a model with constant hashrate and difficulty, deviations like selfish mining [ES14], only increase the relative reward of an actor compared to others and not the absolute reward over time [SSZ16, NKMS16]. So in a constant difficulty model, selfish mining would not be more profitable over time than ordinary mining. This observation also holds in a model with variable difficulty until the difficulty is adjusted. In Bitcoin for example, this happens roughly every two weeks (2016 blocks).

⁹As an analysis of Bitcoin shows [HC21], miners more-or-less stick to the rules, except for preferring transactions with higher fees and smaller blocks for faster propagation

$$\text{EEV}_i(R, \text{HONEST}, n) := n \cdot p_i \quad (6.1)$$

$$\rho_i := \text{EEV}_i(R, \text{HONEST}) - \text{costs}_{\text{mining}} \cdot n \quad (6.2)$$

If the respective actor i also performs and receives transactions, and all of them are uniquely attributable to i , then expected extractable value has to be extended by the extractable value from those transactions. As we are in a scenario where all actors act HONEST, no malicious forks¹⁰ will happen.

Apart from such simple toy examples, estimating the expected extractable value becomes more involved, as soon as different attacks and their probabilities and consequences should be captured. As there are plenty of possible attacks, we cannot cover them all in this thesis and refer to the related research on estimating the success probability in such cases [Ros14, SZ16, ZQT⁺20, GKW⁺16, LK17]. For the rest of the chapter we focus on the potential economic consequences of large-scale attacks and their relation to the minimum EEV (given in normalized block rewards) required to incentivize deviating strategies that could lead to consensus instability while accounting for *all* future rewards.

6.3 Estimation of the EEV in the Context of Attacks

We now look at the question how much EEV for a participant can lead to consensus instability. Thus we have to investigate the EEV in presence of attacks and especially their economic consequences. Therefore, we view the cryptocurrency R from a game-theoretic standpoint and model it as an infinitely repeated game. But first we describe how the EEV can be used to compare different strategies against each other. The question whether any attack strategy is profitable for some actor i , can be summarized by comparing the EEV as well as the costs of the attack against the behavior intended by the protocol designer, i.e., the strategy HONEST, for that actor.

$$\text{EEV}_i(R, \text{ATTACK}) - \text{costs}_{\text{ATTACK}} > \text{EEV}_i(R, \text{HONEST}) - \text{costs}_{\text{HONEST}} \quad (6.3)$$

In other words, if a deviation from the HONEST strategy is more profitable, then this strategy is economically rational. Here the costs can also incorporate potential losses of value of already accumulated resources due to negative consequences of the attack on the exchange rate of those resources. The security and incentive compatibility of a cryptocurrency, against an attack strategy ATTACK, can thus be ensured if the following condition in Formula 6.3 holds at any time for all actors. Formula 6.3 can also be described as a version of the formula provided by Böhme in a presentation [Böh19].

$$u_P(w(P)) - c(P) > u_{\bar{P}}(w(\bar{P})) - c(\bar{P}) + s(\bar{P}) \quad (6.4)$$

Hereby, P denotes the strategy of following the original protocol, whereas \bar{P} stands for the worst of all other actions (attacks). The function $u(\cdot)$ provides the utility and thereby

¹⁰With the simplifying assumption that no blocks are found concurrently.

reflects the real-world preference of an implicit actor which is not explicitly denoted. The function $w(\cdot)$ provides the wealth in protocol coins, which could potentially be reduced by the costs $c(\cdot)$ of launching the attack. There may also be a side-payment (bribe) $s(\cdot)$ to compensate for this loss.

Observation 5. The expected extractable value, as well as the utility of an actor, describe the achievable gain from a certain strategy. Thus the utility of an actor as well as the EEV are equivalent and can be used as synonyms.

In our model a side-payment, or “bribe”, can be expressed as part of the EEV e.g., as an incoming payment that is only valid on the chain desired by the attacker (hence conditional). This illustrates that (side-)payments can influence the incentives of actors. If the EEV of an attack is large enough to overcompensate for the induced costs, an attacker can use a portion of his profit to bribe other miners to convince them to mine on the attack chain. Using side-payments any economically rational actor can be incentivized to support an attack. The question directly related to EEV is: How large does such a side-payment have to be to incentivize illicit activity of other actors. To address this question we also have to take into account potential future EEV (or payoffs/rewards in terms of game theory) and the reduction of such, in case of an event that reduces the exchange rate for the attacked resource, i.e., a value loss.

6.3.1 Single Resource (R)

We now compare potential future EEV and thus the overall payoff for different strategies. From a game-theoretic point of view, we model a cryptocurrency as an infinitely repeated game with discounting¹¹. Therefore, we have to define a *discount factor* $\delta \in (0, 1)$, which specifies the preference of either immediate or future rewards. If δ is close to 0 immediate rewards are preferred. If δ is close to 1 future rewards are almost as good as immediate rewards. If $\delta = 0$ we would have a single-shot game as there would not be any future reward. To account for mining shares and δ , we have to extend our definition of a resource:

Definition 7 (A resource R). From the perspective of an actor i each resource R is a quadruple $\langle r, e, p, \delta \rangle$ consisting of: The quantity r the actor i holds in the respective resource. The exchange rate e for the conversion in value units which reflect the individual preference of that actor. A parameter p which represents the power (e.g., hashrate) of that actor in this resource, which is used together with a discount factor δ to denote expected future rewards in that resource.

¹¹Pass et al. [PS17] pointed out that PoW blockchains cannot stop without becoming insecure, so they have to run infinitely long.

As the payoffs in an infinite game create a geometric series ($p + p \cdot \delta + p \cdot \delta^2 + p \cdot \delta^3 \dots$), the payoff for the first n rounds can be written as:

$$r_n := \frac{p_i \cdot (1 - \delta^n)}{1 - \delta} \quad (6.5)$$

This can be rewritten as a closed form formula for the infinite case since δ^n goes to 0 as n goes to infinity. Thus the EEV for a single actor i with hashrate $p_i \leq 1$ in our infinite game, where every actor plays HONEST, can be approximated by:

$$\text{EEV}_i(R, \text{HONEST}, \infty) := \frac{p_i}{1 - \delta} \quad (6.6)$$

This estimation again denotes the EEV in normalized block rewards as a value unit (with $e = 1$) and assumes that the hashrate of actor i remains static in relation to the hashrates of all other actors.

We now compare this payoff to another strategy which requires a different (attack) action once and then falls back to the original honest behavior, but with a potential negative consequence on future rewards as the exchange rate has dropped. This is comparable to a *grim trigger* strategy in infinitely repeated games, although in our case the environment executes the grim trigger strategy by devaluing the global exchange rate ($e < 1$).

In our scenario, ε is the one-time side-payment to motivate the deviation and e is the value loss in terms of a drop in exchange rate, which of course also has the same negative impact on future EEV and thus must also be accounted for in all potential future mining rewards if the loss is (in the worst case) permanent.

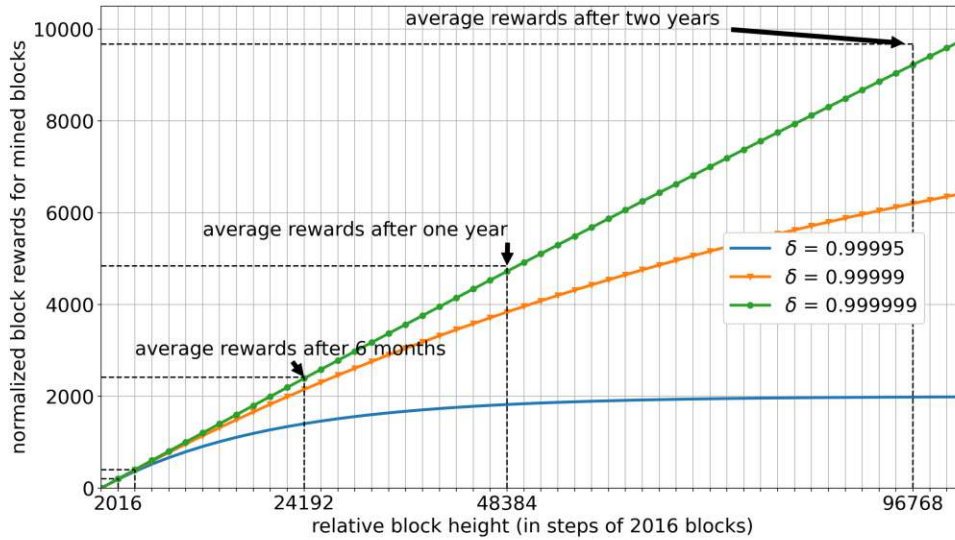
$$\text{EEV}_i(R, \text{ATTACK}, \infty) := \varepsilon + \frac{\delta \cdot p_i \cdot e}{1 - \delta} \quad (6.7)$$

As we are only interested in an approximation, we abstract the particular success probability calculations to evaluate the likelihood of a single attack being successful. Furthermore, we omit the loss of blocks a miner potentially faces if the chain he contributed to becomes stale. If known, this value can be included by adding it to the required bribe ε .

We now estimate how high this one-time side-payment ε has to be to incentivize a one-time deviation from the HONEST strategy with permanent consequence on e for a mainstream cryptocurrency. Therefore, we first have to define some plausible range for the discount factor δ miners might have in practice. Figure 6.1 shows the normalized block reward after a certain number of passed blocks for different values of δ and a hashrate of $p = 0.1$. It can be observed that a relatively high value $\delta = 0.99995$ is needed already to approximate (within a 5% margin) the average income in normalized block rewards after one Bitcoin difficulty period (2016 blocks). For a far sighted miner that has a one to two year interest in Bitcoin a $\delta = 0.999999$ would suffice to be within a margin of 5% of the average number of normalized block rewards after two years.

6. ESTIMATING (M)EV AND ITS CONSEQUENCES

Figure 6.1: Figure to approximate δ by comparing the average block rewards received by a miner with $p = 0.1$, to the expected infinite game rewards for the same miner with different discount factors δ . All rewards are given in normalized block rewards.



Now that we have picked some plausible values for δ , we can approximate the required total side-payment ε that would be required to change the incentives of participating miners with different hashrates. Therefore, we compare the EEV of honest behavior with the EEV of the attack. Assuming the costs of mining in both cases are identical, the EEV of the attack has to be more profitable than the honest behavior, for the attack to be realistic.

So far we have not taken into account that miners could also hold funds ($f_i(r)$), which are distinct from hashrate (which describes future gains given out as currency units). Since a successful attack will lead to a potential drop in the exchange rate, we have to consider this for all future rewards, as well as all funds the miner is currently holding. Equation 6.10 compares the two strategies under the assumption that there is only one resource (r) the respective miner cares about. Hereby, the hashrate is viewed as some share in the protocol which provides future rewards in the respective cryptocurrency (in r) proportional to the size of the share.

$$EEV_i(R, \text{HONEST}, \infty) := \frac{p}{1-\delta} + f_i(r) \quad (6.8)$$

$$EEV_i(R, \text{ATTACK}, \infty) := \varepsilon + \frac{\delta \cdot p \cdot e}{1-\delta} + f_i(r, e) \quad (6.9)$$

$$EEV_i(R, \text{HONEST}, \infty) < EEV_i(R, \text{ATTACK}, \infty) \quad (6.10)$$

Solving equation 6.10 for ε , we can calculate the required side-payment for the following example: To compensate a five percent value drop ($e = 0.95$) for a miner with zero funds ($f_i(r) = 0$) and 10% hashrate ($p = 0.1$), a side-payment in approximately the size of 500 times the normalized block reward is needed (if $\delta = 0.99999$). If the side-payment itself is performed in r , and thus subject to the same value drop of 5% as well, then ≈ 527 times the normalized block reward is required as a side-payment.

Although theoretically possible, such high bribes in the size of hundreds of normalized block rewards appear unlikely in practice from a current stand point. Moreover, for an attack to be economically viable for an attacker, he would have to perform a double-spend of a transaction which is much larger than the required overall side-payments. Ideally the attacker himself (as well as the victim) does not possess any hashrate in the targeted cryptocurrency, such that his personal future income will not be negatively affected by the consequences of the attack, i.e., drop in exchange rate. Moreover, the attacker is advised to use all his funds in r in the double-spend transaction to further minimize the negative effects on the exchange rate. The leftovers from the double-spend, after subtracting the required side-payments to incentivize a sufficient portion of the hashrate to support the attack chain, could be viewed as profit for the attacker. So for such an attack to work, funds would have to be unevenly distributed amongst actors and an individual payment must only be limited by the available overall supply of the respective resource. Then the amount of a double-spend can theoretically be high enough that the excess profit of the attacker can be used to bribe a majority of miners to support an attack chain.

Observation 6. In a scenario where there is only one cryptocurrency participants care about, the side-payment necessary to incentivize a deviation has to account for all current and future losses.

6.3.2 Multiple Resources (\mathcal{R})

The analysis so far assumed that all actors only care about the same single resource, i.e., cryptocurrency, and express their extractable value in normalized block rewards of this resource. The resulting question is, what if there are multiple resources and not all actors necessarily care about the same set of resources to a comparable degree?

To approach this question, we modify equation 6.10 to estimate the EEV for the honest strategy, as well as for the attack strategy, by accounting for all resources (e.g.,

cryptocurrencies) a player i cares about. Hereby, we model the individual hashrates (p) as a part of each resource which defines the share of future rewards an actor will receive in the respective resource. Additionally there is a set of δ values for each resource. Moreover, in this scenario a bribe does not necessarily have to be paid in the resource where the attack action should happen, thus several bribes are possible $\{\varepsilon_0, \varepsilon_1, \varepsilon_2, \dots\}$.

$$\text{EEV}_i(\mathcal{R}, \text{HONEST}, \infty) := \sum_{j=0}^{|\mathcal{R}|} \left(\frac{p_j}{1 - \delta_j} \right) + \sum_{j=0}^{|\mathcal{R}|} f_i(r_j) \quad (6.11)$$

$$\text{EEV}_i(\mathcal{R}, \text{ATTACK}, \infty) := f_i(r_0 + \varepsilon_0, e_0) + \frac{\delta_0 \cdot p_0 \cdot e_0}{1 - \delta_0} \quad (6.12)$$

$$+ \sum_{j=1}^{|\mathcal{R}|} \left(\frac{p_j \cdot e_j}{1 - \delta_j} \right) + \sum_{j=1}^{|\mathcal{R}|} f_i(r_j + \varepsilon_j, e_j) \quad (6.13)$$

Compared to the single resource case in Section 6.3.1 the multi-resource case now allows actors to escape certain negative consequences on the exchange rate e_0 , e.g., by moving their hashrate to another permissionless PoW cryptocurrency (like for example R_1). This of course only works if the miner's hardware can also be efficiently used in the other cryptocurrency and e_1 is not affected to the same degree, or generally much higher to begin with ($e_1 \geq e_0$ and $\delta_1 \geq \delta_0$). If also current holdings in resources can be transferred to other resources through exchange services, then in the worst case it may be possible to evade all negative consequences from attacks on a certain target resource R_0 . To which degree such negative consequences can be evaded depends on several factors: The availability of adequate alternatives, the type of the attack, as well as how fast resources can be moved and exchange rates adopt (cf. [BBTL22]).

If such evasion techniques are possible, this raises an interesting question from a game-theoretic point of view. The previously infinitely repeated game, now becomes a finite game, as actors can leave the system at will. Therefore, the option to defect in the (personally) last round of the (now) finite game suddenly becomes an economically rational strategy.

Observation 7. If appropriate alternative resources exist, parties can evade negative attack consequences on their overall EEV, by moving their assets to another less, or even positively affected resource. Thereby, the once infinite game becomes finite from their perspective.

6.4 Example Scenarios with Multiple Resources

In the following we describe and visualize some hypothetical profitable and non-profitable events in this multi-resource model from the perspective of an individual player. Therefore, we compare the normalized EEV before the event with the EEV after the event. To denote the difference of an event not in absolute numbers, but relative to the current

valuation of some miner or user, we assume that the normalized valuation includes all expected future payoffs, as well as all currently held funds before the event happens. We then calculate the difference to this value after the event. Equation 6.14 shows this simplified relation for an event to be profitable compared to the current state.

$$\sum_{j=0}^{|\mathcal{R}_i|} e_j \cdot (r_j + p_j) < \sum_{j=0}^{|\mathcal{R}'_i|} e'_j \cdot (r'_j + p'_j) \quad (6.14)$$

Figure 6.2 and 6.3 provide visualizations of the consequences of the relative difference in the valuation of the player in terms of his desired resources. Each sub-figure visually compares the total normalized valuation of the player before an hypothetical event, with the valuation of the same actor after the respective event.

Double-spend: Figure 6.2a shows a classic double-spend attack on a cryptocurrency specified as resource $R_0 = \langle r_0, p_0 \rangle$ for the respective player. This player is also invested in another resource $R_1 = \langle r_1, p_1 \rangle$, where the total valuation of the player is normalized. This means all his current and expected future holdings, that she will earn with her hashrate/voting power, in the respective cryptocurrencies is given by $r_0 + r_1 + p_0 + p_1 = 1$, where $r_0 = 0.1, p_0 = 0.2$ and $r_1 = 0.2, p_1 = 0.5$. In other words the player is more invested into R_1 than in R_0 . If the player now decides to launch a double-spending attack on R_0 , this can lead to a drop in the exchange rate of this cryptocurrency (e_0), but might leave the exchange rate e_1 of the other resource unaffected. If the double-spend is successful, the original funds in R_1 can be spend again, but the attacker has already received the purchased good, hows value is denoted with r_3 in this case, as it might be denominated in an additional resource. Additionally, if the voting power (hashrate p_0) can be switched to R_1 , the negative effects on the exchange rate on the hashing power p_0 can be evaded. In summary, the example attack is profitable even if the held funds r_0 cannot be transferred in this example parameterization.

Goldfinger attack: Figure 6.2b shows a Goldfinger attack on a cryptocurrency R_0 . In this scenario the player is invested more in R_1 compared to R_0 , therefore the drop in exchange rate $e'_0 < e_0$ is compensated by the increase of exchange rate in the other resource, i.e., $e'_1 > e_1$. Due to the distribution of funds and mining resources the Goldfinger attack results in an overall gain for this specific player.

Chain split from the perspective of a miner: Figure 6.2c shows a permanent hard-fork, or chain split, in a PoW cryptocurrency from the perspective of a miner. The total valuation (funds + expected future payoffs), i.e., the EEV is normalized to one before the fork ($e_0 = 1$). In this example, 85% come from expected mining rewards, and 15% from current holdings in r_0 . After the fork the 15% are suddenly also available in another cryptocurrency r_1 , but the exchange rate of r_0 has dropped to $e'_0 = 0.5$ and the exchange rate for the newly created cryptocurrency is given by $e_1 = 0.7$. Thus, overall the miners loses more than he gains from the chain split.

Chain split from the perspective of a user: Figure 6.2d shows the same situation as in Figure 6.2c but now from the perspective of a user who now also holds funds in r_1 after the fork. Due to the fork, his new funds in r_1 are worth 70% of the funds he has in r_0 . Therefore, from the perspective of this user a fork is always profitable if the exchange rates of both cryptocurrencies combined is higher than the original exchange rate of the hard-forked cryptocurrency ($e'_0 + e_1 > e_0$).

Attack without escape resource: Figure 6.3a depicts an attack on a cryptocurrency, again from the perspective of a miner. In this case 80% ($p_0 = 0.8$) of the miners future expected rewards come from mining, while at the same time the 20% of this current valuation come from held funds. If there now is an attack in which the value of r_0 drops ($e_0 = 0.9$), this of course would have a negative effect on the valuation of this miner. Our example shows a total value loss of 10% ($e = 0.9$) in r_0 , leading to an overall drop of the miners current valuation below 1.

Attack with escape route: Figure 6.3b shows the same situation, but now the miner can move his funds into a different cryptocurrency r_1 , which does not suffer from a value loss ($e_1 = e'_1 = 1$). Thereby, the induced financial damage can be reduced. In this case the hashrate of the miner cannot be moved to another cryptocurrency, because there is no profitable alternative which uses the same PoW algorithm, i.e., the specialized hardware is bound to R_0 . To over compensate for the remaining losses from future mining rewards in R_0 , the miner receives a bribe in r_1 . Thereby, the overall valuation of the miner increases thus making the underlying actions economically rational from the perspective of this particular miner.

Merged mining: Figure 6.3c shows the opportunity to merge mine a different cryptocurrency with the same hashrate p_0 , but a much lower exchange rate $e_1 < e_0$. Since, in this case, merged mining has no negative effect on the exchange rate of R_0 , any additional gain from utilizing the already available hashrate would be profitable for the player in this model. Of course, in practice the additional costs of adapting the software accordingly has to be taken into account.

Pitchforks attack: Figure 6.3d shows a pitchfork attack, in which the same hashrate as well as the funds are suddenly available in another newly created cryptocurrency, but the original cryptocurrency faces a drop in exchange rate ($e_0 < e'_0$). If this drop in exchange rate is smaller than the exchange rate of the newly created cryptocurrency then such an attack would be profitable. This assumes, that the mining strategy that has to be applied to blocks, such that they are valid in both cryptocurrencies, has not negative impact on the income in R_0 . If this is the case, this negative consequence has to be accounted for as well.

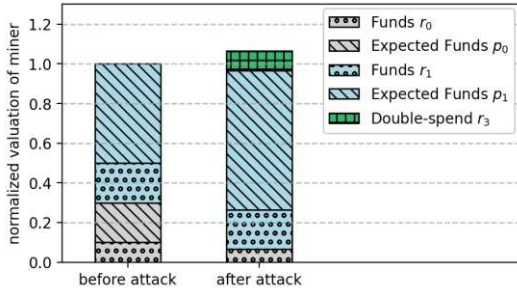
These examples further illustrate, that miners who are tied to a cryptocurrency due to their specific mining hardware, have a higher incentive not to risk negative consequences

on the exchange rate of that cryptocurrency. If switching to another equally, or more profitable alternative is possible though, attacks become more attractive. This highlights, that in an environment in which multiple cryptocurrencies co-exist and represent alternative resources to each other, the EEV cannot be estimated by looking at a single resource/cryptocurrency alone. Especially since cryptocurrencies can be created at any point in time, for example through forks, which change the overall cryptocurrency landscape and potentially affect the exchange rates of existing cryptocurrencies in one, or the other way.

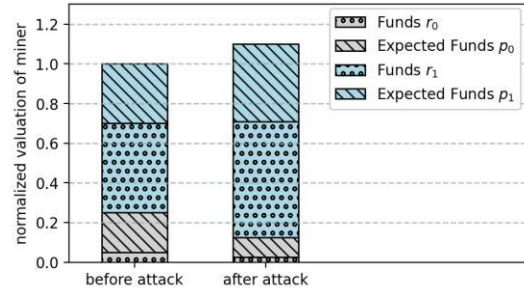
Observation 8. In the multi-cryptocurrency environment where new resources can be created, the EEV can be influenced by these new resources, since the set of available resources for actors changes. Thereby, providing new alternatives, or modifying existing exchange rates and discount factors.

This problem of considering out-of-band income streams in economic security models of permissionless PoW cryptocurrencies, is also nicely illustrated by various bribing, or algorithmic incentive manipulation attacks which utilize out-of-band payments [Bon16, TJS16, MHM18, JSZ⁺19, JSZ⁺21b] as well as other economic arguments regarding the incentive structure of such systems [FB19, Bud18].

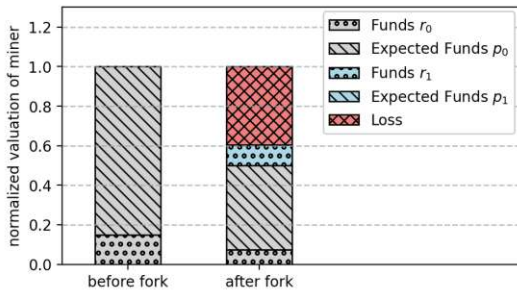
Figure 6.2: Visual illustration of different events and their consequences on a participant with $\mathcal{R} := \{R_0, R_1\}$. The total valuation of a participant *before* the respective event is normalized to 1. This means that the values for the initial exchange rates are $e_{0,1} = 1$ (s.t. $f(r_{0,1}, 1) = r_{0,1}$), and that δ is static and thus ignored ($\delta_{0,1} = 0$). In other words expected future rewards were already accounted for in the relation between $p_{0,1}$ and $r_{0,1}$, s.t. $p_0 + r_0 + p_1 + r_1 = 1$.



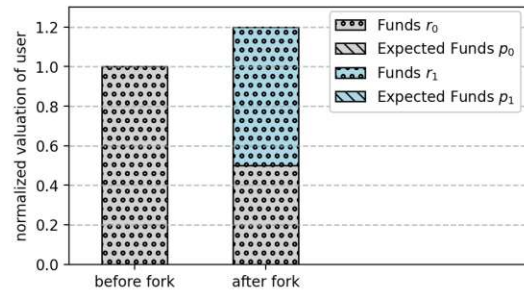
(a) A double-spend attack on a cryptocurrency R_0 from the perspective of a miner. *Before*: The total expected future income from mining is $p_0 = 0.2$ and $p_1 = 0.5$, in relation to the currently held funds in $r_0 = 0.1$ and $r_1 = 0.2$. *After*: A double-spend of all available funds r_0 , leads to an additional gain of $r_3 = 0.1$ through the double-spend, but also to negative consequences on the exchange rate $e'_0 = 0.65$. This drop is evaded by moving all hashrate to R_1 , where the exchange rate remains constant $e'_1 = 1$. This leads to a gain of exactly the exchange rate in R_0 as the double-spend funds cannot be moved without losses: $r_0 \cdot e'_0 + (p_0 + p_1 + r_1) \cdot e'_1 + r_3 = 1.065$



(b) A Goldfinger attack [KDF13, MHM18, Bon18] on a cryptocurrency R_0 from the perspective of a miner. *Before*: The total expected future income from mining is $p_0 = 0.2$ and $p_1 = 0.3$, in relation to the currently held funds in $r_0 = 0.05$ and $r_1 = 0.45$. *After*: The Goldfinger attack leads to a drop in the exchnage rate in R_0 of 50% ($e'_0 = 0.5$), while the exchange rate in R_1 increases by 30% ($e'_1 = 1.3$). Due to the distribution of his current holdings and expected future income in those two resources R_1 and R_2 , at the end of the day m profits more from the increase than he loses from the decrease: $(p_0 + r_0) \cdot e'_0 + (p_1 + r_1) \cdot e'_1 = 1.1$

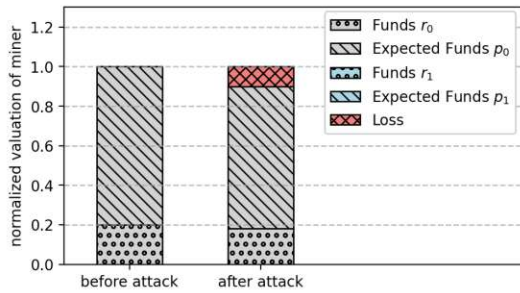


(c) A fork of R_0 into R_0 and R_1 from the perspective of a miner. *Before*: The total expected future mining rewards are $p_0 = 0.85$ while $r_0 = 0.15$ come from current holdings. *After*: The fork changes the exchange rate to $e'_0 = 0.5$, thus cutting the previous total valuation in half and at the same time adds the funds from the new resource $r_1 = r_0 = 0.15$ with an exchange rate $e'_1 = 0.7$. This leads to an overall loss for the miner: $(p_0 + r_0) \cdot e'_0 + r_1 \cdot e'_1 = 0.605$

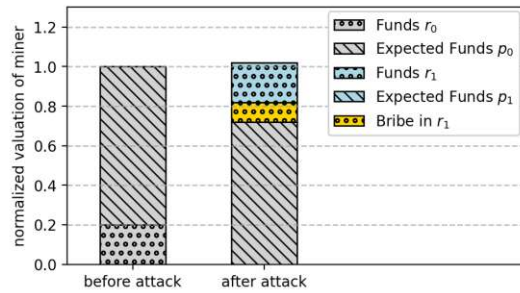


(d) The same fork as in Figure 6.2c from the perspective of a user. *Before*: There is no expected future income $p_0 = 0$, thus 100% come from current holdings in R_0 ($r_0 = 1$). *After*: The forks changes the exchange rate to $e'_0 = 0.5$, thus cutting the previous total valuation in half, but at the same time adds the funds from the new resource $r_1 = r_0 = 1$ with an exchange rate $e_1 = 0.7$. This leads to a surplus of 0.2 for the user in this case: $(p_0 + r_0) \cdot e'_0 + r_1 \cdot e'_1 = 1.2$

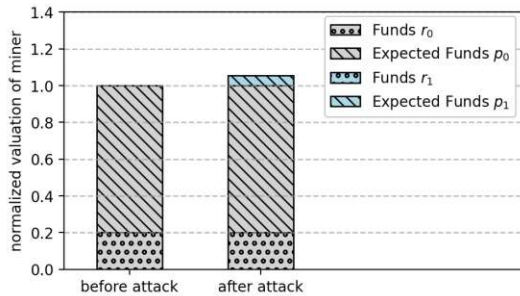
Figure 6.3: Visual illustration of different events and their consequences on a participant with $\mathcal{R} := \{R_0, R_1\}$. The total valuation of a participant *before* the respective event is normalized to 1. This means that the values for the initial exchange rates are $e_{0,1} = 1$ (s.t. $f(r_{0,1}, 1) = r_{0,1}$), and that δ is static and thus ignored ($\delta_{0,1} = 0$). In other words expected future rewards were already accounted for in the relation between $p_{0,1}$ and $r_{0,1}$, s.t. $p_0 + r_0 + p_1 + r_1 = 1$.



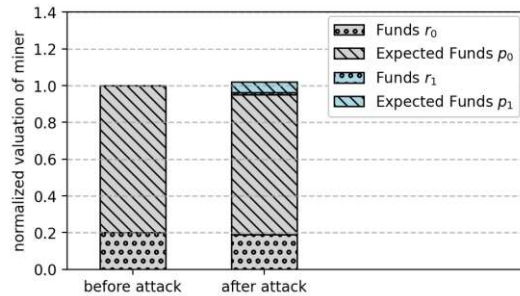
(a) An attack on a R_0 from the perspective of a miner. *Before:* The total expected future income from mining is $p_0 = 0.8$, while $r_0 = 0.2$ come from current holdings. *After:* An attack with negative consequences on the exchange rate $e'_0 = 0.9$ is depicted leading to a loss for the miner:
 $(p_0 + r_0) \cdot e'_0 = 0.9$



(b) The same attack as Figure 6.3a, but in this case the funds $r_0 = 0.2$ can be transferred to an alternative cryptocurrency R_1 in which no value loss occurs ($e'_1 = 1$). Furthermore, a bribe $\varepsilon = 0.1$ is payed in r_1 to the miner to accommodate for the induced losses. In this case the miner would have surplus, although his hashrate p_0 is non-transferable.
 $(p_0 + r_0) \cdot e'_0 + (r_1 + \varepsilon) \cdot e'_1 = 1.02$



(c) An opportunity to merge mine another cryptocurrency R_1 which is worth 7% of R_0 . *Before:* The total expected future income from mining is $p_0 = 0.8$, while $r_0 = 0.2$ come from current holdings. *After:* An additional cryptocurrency is merge mined with the same hashrate and no consequences on the exchange rate of the original one
 $(p_0 + r_0) + (p_1 \cdot e_1) = 1.056$



(d) A pitchfork attack is executed on R_0 which leads to a value drop of 5% s.t. $e'_0 = 0.95$. At the same time the exchange rate of the newly created and merge mined attack coin is given by a slightly higher rate $e_1 = 0.07$, which would result in overall additional gain when joining the pitchfork attack: $(p_0 + r_0) \cdot e'_0 + (p_1 + r_1) \cdot e_1 = 1.02$

6.5 Consequences of Optimizing EEV and Mitigation Strategies

The presented observations highlight that accurately estimating the EEV of a particular miner is impossible, even when knowing all transactions belonging to this miner, as well as all current preferences regarding cryptocurrencies and resources the respective miner cares about and to which degree. There are two major reasons for this: First, it is not possible to predict the actions taken by other actors interacting with the system by issuing transactions which either directly, or indirectly affect the respective miner i , or the exchange rate of a resource. Second, if miner i is open for accepting payments or new resources (e.g., validator roles which provide future incomes), then the fact that new cryptocurrencies can be forked, or created at any point in time (with a free choice of rules and distribution of funds) provides new possibilities of income to i . As even the sheer existence of new cryptocurrencies can have a negative affect on the valuation of existing cryptocurrencies, this can also influence the EEV of i . Even more so, if the rules of the newly created cryptocurrencies are designed in a way that actively harm existing ones, as outlined in [JSSW18]. Algorithmic incentive manipulation (AIM) attacks, with their offered bribes, are another way of how an attacker can interfere with the EEV of players [JSZ⁺21b, JSZ⁺19], providing another potential source of uncertainty regarding the EEV of other actors. In other words, precisely calculating the EEV of a miner i is impossible.

Nevertheless, it may still be possible to approximate the EEV, if the number of possibly available resources \mathcal{R} , the computational capabilities of actors, as well as their overall number, can be meaningfully bounded. As cryptocurrencies provide the possibility to virtually create new resources at any point in time, this can technically not be prevented in practice. It is therefore questionable, if economic security models of permissionless cryptocurrencies that take the interplay between multiple resources into account, will be able to produce satisfactory security guarantees, compared to the high standards regarding security proofs that we are used to, for our cryptographic primitives, formally verified smart contracts, or classical Byzantine fault tolerant consensus systems.

This leaves us with the open question on how to best include economic considerations into the choice of the security parameter k determining the number of required confirmation blocks. In Section 6.5.1 we show a simple workaround that relieves us from the burden of correctly determining the right value for k .

6.5.1 The Let's Go Shopping Defense

We now describe a simple defensive strategy a merchant M can use to transfer the risk of choosing an insufficient security parameter k_M , to another merchant W . In our scenario we, assume that merchant M offers some quantity v of resource r for sale to a customer C . For this technique to work, we need to assume that there exists a merchant W with

an already defined security parameter k_W . Furthermore, merchant W offers some easily tradable good/resource r' that is purchasable in arbitrary quantity and also stable in price. Then merchant M can now choose his k_M such that $k_M > k_W$, and immediately use all received funds directly to acquire r' . Therefore, M has to create a transaction tx_M that immediately uses all funds that were used by the customer to purchase r . In other words the transaction of M builds up on the transaction of C , i.e., $tx_C \rightarrow tx_M$. In an UTXO model cryptocurrency this can be achieved by using the respective UTXO as input, whereas in an account-based model a dedicated account can be created to handle the purchase¹². Technically, in most prevalent cryptocurrencies tx_C and tx_M can even be part of the same block if included in the right order and if M broadcasts tx_M immediately after observing tx_C in the P2P network.

Using this technique, merchant M can be sure to receive $v_{r'}$ before he has to hand out v_r . If now transaction tx_C is double-spent, or otherwise is invalidated so will be tx_M , but at that point either M has not yet handed out v_r , or already received $v_{r'}$. In both cases M does not face any direct damage from an attack.

6.5.2 Consequences of (Miner) Extractable Value

We have shown that MEV is a special form of value extractable by miners, and that there is a difference between the *extractable value* that is computed in retrospect and *expected extractable value* (EEV) that is also forward looking and takes the probability of events influencing it into account. Further we have shown that estimating the EEV of any actor i is hard or even impossible in practice as we have to deal with imperfect information and possibly multiple cryptographically interlinked cryptocurrencies. The EEV depends on several factors: Transactions affecting i , which reach from incoming and outgoing regular payments, over bribes, to front-running and arbitrage opportunities, as well as all consequences of actions affecting the valuation of assets in different resources actor i cares about. The difficulty to accurately estimate the EEV is further amplified by the fact that new cryptocurrencies might pop up, increasing the set of resources actors care about, or putting pressure on existing cryptocurrencies. Although, theoretically workable, a rather unsatisfactory workaround is described to transfer the risk of choosing an insufficiently large security parameter k to another merchant. In the wake of more and more attacks that exploit aspects of the economic rationality of actors (like for example front-running), a better understanding of the economic interplay between such actors, as well as cryptocurrency systems as a whole, is desperately required to more accurately model the security guarantees of prevalent permissionless cryptocurrencies under such economical considerations and attacks. If the lack of descriptive models (which take practical economic considerations into account) persists, we have to ask ourselves if economic incentives in permissionless cryptocurrencies can ever produce satisfactory security grantees, our just occasionally worked “better in practice, than in theory” for a while.

¹²Alternatively a smart contract can be used to execute any future trade immediately, but we ignore this possibility for now.

Adversarial Games: Formalization of AIM attacks on Nakamoto Consensus

In Chapter 6, we have shown that the concept of AIM using bribes and extractable value (EEV) are related since a bribe can be viewed as another source of extractable value. In this chapter, we are interested in the conditions when AIM attacks are possible from a technical standpoint, as well as highlighting the underlying assumptions that are required in this regard. To *express and explain attacks* on Nakamoto consensus-based permissionless cryptocurrencies, such as front-running, double-spending, or bribing in general, under a common technical framework, we define a state machine replication (SMR) based model tailored for prevalent permissionless cryptocurrencies. This should improve the understanding of these attacks, which are regularly observed in practice, and make different types of attacks more comparable. Thereby, the security of such protocols can be assessed more accurately while taking economic considerations into account. Moreover, our model allows us to recap and classify different extractable value optimization approaches and attacks, as well as different types of temporary and permanent forks under a common framework. In the end, the presented model is used to show that the creation of adversarial games is always possible from a technical standpoint. Assuming that the targeted Nakamoto consensus-based permissionless cryptocurrency is efficiently verifiable and eventually makes progress.

7.1 Nakamoto Consensus-based State Machine Replication

In our model, we try to relate certain aspects and design decisions of permissionless cryptocurrencies based on Nakamoto consensus (NC) to the area of distributed systems, especially state machine replication (SMR) [CSV16]. Thereby, we also want to take certain economic considerations and incentives into account. The main goal of our model is to express and explain observed attacks such as front-running, double-spending, or bribing in general under a common framework. Therefore, our model of Nakamoto consensus-based state machine replication should resemble the actual functionality of mainstream cryptocurrencies as closely as possible and, at the same time, overlap as much as possible with established formalisms used to reason about such protocols, such as the *backbone model* [GKL20] or subsequent security analysis of NC-style protocols [GKR20, DKT⁺20].

To better understand and describe the characteristics of known attacks with our model, it is necessary to distinguish between *operations* that perform the actual state transitions within the system and *transactions* which represent the atomic execution steps describing these operations. In NC-style cryptocurrencies, the operations which advance the state of the system are called *blocks*. Unlike classical operations in the context of state machine replications, blocks additionally contain a cryptographic reference to their respective predecessor. Moreover, the validity of a block cannot be determined independently but is dependent on the history of its predecessors. Therefore, the validity of a single transaction cannot be determined independently as well because it is also dependent on the transactions which happened before. Another distinguishing characteristic is that transactions cannot be created arbitrarily but require a cryptographic signature of an authorized party. In other words, in NC-style cryptocurrencies, transactions are batched together into *blocks* which form a *blockchain* representing the history of performed state transitions in the context of this chain.

Definition 8 (Operation o). An operation is a finite ordered sequence of atomic execution steps of length at least 1 and at most \aleph describing a state transition and optionally some additional metadata.

A state is subsequently defined by the sequence of operations leading to this point (see 17).

Definition 9 (Transaction tx). A transaction is a finite description of an atomic execution step within an operation together with a cryptographic signature.

The details of how a transaction encodes an atomic execution step are not important for our model. Important is that by our definitions:

- Transactions are not allowed to be infinitely large, and therefore, there can only be finitely many possible transactions.

- Transactions must comply with formatting rules checked by the predicate $\text{WELLFORMED}(\cdot)$ (see Section 7.1 for details).
- The validity of a single transaction cannot be determined individually but requires the context of all previous transactions, i.e., a system state, which is determined by the execution of these transactions and the associated execution function $\text{EXECUTE}(\cdot)$ (see Section 18 for details), which represents the interpretation rules of the respective protocol.

An unordered set of transactions is denoted by \mathcal{T} , e.g., $\{tx_2, tx_0, tx_1\}$, while an ordered sequence of transactions is denoted by τ , e.g., $\tau = (tx_0, tx_1, tx_2)$. The ordering of a sequence of transactions is not necessarily dependent on some property of the respective transactions. If an ordered sequence of transactions τ exists as a subsequence in another ordered sequence of transactions τ' , we can write $\tau \prec \tau'$.

Since we focus on blockchains and cryptocurrencies, *all operations in our model are assumed to be blocks*. The key difference is that all blocks, except the genesis block, contain a cryptographic reference to their predecessor.

Definition 10 (Block b_n). A block is an operation with a cryptographic reference to its predecessor. It is a finite ordered sequence of transactions of length at least 1 and at most \aleph describing a state transition, as well as a cryptographic hash which identifies the previous operation $n - 1$, or no cryptographic hash iff $n = 1$ and represents the first block in a chain, also called genesis block.

Definition 11 (Blockchain c_n). A blockchain $c_n = (b_1, b_2, \dots, b_n)$ is a finite ordered sequence of $n = |c_n|$ cryptographically linked blocks starting from an initially defined genesis block b_1 with no predecessor. If the length of the chain is not relevant, the subscript is omitted.

Analogous to our notation for sequences of transactions, we write $c \prec c'$ if a sequence of blocks c exists somewhere in the prefix of another sequence of blocks c' . If a block b exists somewhere in a chain c , we write $b \in c$. A single coherent blockchain can only have one block at any given height. However, there can exist multiple blockchains with overlapping blocks in their prefix. In this case, the result is an *in-tree* of blocks, which is described in more detail later in this section. A coherent blockchain represents a single path to the root (genesis block) in this *in-tree*.

We further define the following helper functions, which return the length of a given chain, the last block in a given chain, and the sequence of transactions from a block or sequence of blocks.

Definition 12 ($\text{LENGTH}(c)$ or $|c|$). The function $\text{LENGTH}(c)$, abbreviated as $|c|$, returns the number of blocks, i.e., the length, of the given chain.

Definition 13 ($\text{HEAD}(c)$). The function $\text{HEAD}(c)$ returns the last block, i.e., the tip, in a given chain.

Definition 14 ($\text{PREV}(b_n)$). The function $\text{PREV}(b_n)$ returns the previous block, i.e., b_{n-1} , in the given chain.

Definition 15 ($\text{CHAIN}(b_n)$). The function $\text{CHAIN}(\cdot)$ returns the entire ordered sequence of blocks (i.e., the blockchain), which consists of all predecessors leading to the given block.

Definition 16 ($\text{TXSEQ}(b_n)$ or $\text{TXSEQ}(c)$). The function $\text{TXSEQ}(\cdot)$ returns the ordered sequence of transactions that have been included in the given block (b_n) or chain (c).

After defining the required data structures, we can define the concept of a state.

Definition 17 (State s_n). We define a system state s_n as the interpretation of a finite ordered sequence of n operations determined by its execution and an initial starting state s_1 which in turn is determined by a genesis block b_1 .

Based on the definition of *state machine replication* (SMR) by Cachin et al. [CSV16], we define an execution function as follows.

Definition 18 ($\text{EXECUTE}(s_n, b_{n+1})$). We define a deterministic execution function that takes a starting state s together with an operation¹ b as input and outputs a successor state s' together with some response or result rst . The genesis block b_1 defines the initial state s_1 and therefore represents the only transition with no predecessor state, i.e., $s_0 = \perp$. The function is required to terminate on all inputs. Moreover, we require that the function runs in a polynomially bounded number of computational steps. If an error occurs or no subsequent state can be computed after this upper bound of computational steps, the original state is returned, and $rst := \perp$.

$$\text{EXECUTE}(s_n, b_{n+1}) := \begin{cases} \text{EXECUTE}(s_0, b_1) \wedge s_0 = \perp & \langle s_1, rst_1 \rangle \\ \text{EXECUTE}(s_n, b_{n+1}) \text{ where } b_{n+1} \text{ leads to a new state} & \langle s_{n+1}, rst_{n+1} \rangle \\ \text{EXECUTE}(s_n, b'_{n+1}) \text{ where } b'_{n+1} \text{ does not lead to a new state} & \langle s_n, \perp \rangle \end{cases}$$

This ensures that when a correct process executes a sequence of operations (b_1, \dots, b_n) , then the sequences of states (s_0, s_1, \dots, s_n) and responses (rst_1, \dots, rst_n) , i.e., outputs, satisfy $\forall i \in (1, \dots, n) \text{EXECUTE}(s_{i-1}, b_i) = \langle s_i, rst_i \rangle$.

Given our definition of the execution function, we can now define the validity of a block by also requiring the validity of all blocks before, which led to this particular block. The validity of the genesis block as the first block (b_1) in any chain is given by definition, as this block determines the starting state.

¹In our case all operations are assumed to be blocks as they contain a cryptographic reference to their predecessor.

Definition 19 ($\text{VALID}(b_n)$). We define a predicate $\text{VALID}(b_n) \rightarrow \{\text{TRUE}, \text{FALSE}\}$, which returns true when a given operation (b_n) is valid according to the rules of the protocol, false otherwise. The validity of a block b_n can only be determined by looking at the respective blockchain, i.e., the sequence of operations till b_n . Therefore, the predicate for blocks is recursively defined starting from a genesis block b_1 which is valid per definition:

$$\text{VALID}(b_n) := \begin{cases} b_n = b_1 & \text{TRUE} \\ \text{VALID}(b_{n-1}) \wedge \text{EXECUTE}(s_{n-1}, b_n) = \langle s_n, rst_n \rangle \wedge rst_n \neq \perp & \text{TRUE} \\ \text{otherwise} & \text{FALSE} \end{cases}$$

Definition 20 (Set of potentially constructible valid blocks \mathcal{O}). We define the set of all potentially constructible valid blocks, starting from the genesis block b_1 of the protocol, as the set of finite input strings from the domain \mathcal{D} , for which the predicate $\text{VALID}(b)$ returns true.

$$\mathcal{O} := \{b \in \mathcal{D} \mid \text{VALID}(b)\}$$

Although permissionless PoW blockchains are theoretically required to run infinitely long [PS17], in practice, there will only ever exist chains of finite length. Therefore, we restrict our model to only consider instances of protocol runs with finitely many blocks. See Section 7.1.2 for more Information.

Definition 21 (Set of potentially reachable states \mathcal{S}). We define the set of all potentially reachable system states \mathcal{S} , over the domain of potentially valid blocks \mathcal{O} , as the set of states that could be reached starting from an initial state of the protocol determined by the genesis block b_1 , when given as input to the function $\text{EXECUTE}(\cdot, \cdot)$, i.e.,

$$\mathcal{S} := \{b_n \in \mathcal{O} \left[\bigcup_{i=1}^n \text{EXECUTE}(s_{i-1}, b_i) \right]\}$$

To extend a chain, a new block has to be created. In the context of PoW this process is referred to as mining performed by a miner, whereas in the context of PoS a new block is signed by an authorized proposer/validator. To account for different system rules under which a new valid block can be created, we define a function $\text{PROPOSE}(\cdot, \cdot)$, which extends a sequence by a new valid block. This function takes the block to extend upon b_n and a block template \bar{b}_{n+1} of its successor. The block template contains all transactions but is missing a valid system-specific approval, such as a PoW, or a cryptographic signature by one or more validators in PoS.

Definition 22 ($\text{PROPOSE}(b_n, \bar{b}_{n+1})$). We define a proposing function that takes a predecessor block b_n , upon which to extend, as well as a template for a new block \bar{b}_{n+1} and returns a new valid block b_{n+1} , s.t. $\text{VALID}(b_{n+1})$, or $\{\emptyset\}$ if no valid block can be produced from the given template.

Equation 7.3 shows an example application of the mining function in our notation. The chain containing a not yet rightfully proposed block (\bar{b}_4) is also marked with a bar (\bar{c}_4):

$$c_3 := (b_1, b_2, b_3) \quad (7.1)$$

$$\bar{c}_4 := (b_1, b_2, b_3, \bar{b}_4) \quad (7.2)$$

$$c_4 := (b_1, b_2, b_3, \text{PROPOSE}(b_3, \bar{b}_4)) \quad (7.3)$$

$$c_4 = (b_1, b_2, b_3, b_4) \quad (7.4)$$

The longer the sequence of blocks (i.e., a blockchain) that builds up on a particular block, the higher the confidence that this particular block indeed represents some agreed-upon state of the system confirming the transactions within it. To reason about different chains of blocks and detect and agree on a common prefix, we need a function to identify the *canonical chain*, also sometimes referred to as the *main chain* when viewed from an individual perspective of a participant. For example, in a PoW-based model with constant difficulty (i.e., all blocks have the same difficulty target), the longest chain is also the heaviest chain with the most accumulated PoW and, thus, the canonical chain. We define a function $\text{MAIN}(\cdot)$, which returns the main chain given a graph containing all chains of blocks known by the respective participant. In our case this data structure is a rooted directed acyclic graph (DAG), where each node has exactly one outgoing edge. More accurately, the set of possible blockchains starting at a certain genesis block form an *anti-arborescence* (also referred to as *in-tree*). An arborescence is a directed, rooted tree in which all edges point away from the root. Additionally, for every vertex there is precisely one directed path to the root. An anti-arborescence is an arborescence that converges to a root node (in our case, the genesis block) instead of diverging from it. We denote this data structure with *InTree*.

Finding the main chain (where all blocks have the same difficulty target) corresponds to the problem of finding the longest path in the given graph. The problem of finding the longest path in a graph is NP-hard in general, but the complexity of finding the longest path in a DAG is linear in the number of vertices and edges of the DAG [CLRS01], i.e., $O(|V| + |E|)$, and the maximum number of edges in a DAG is bounded by $\sum_{i=1}^n i = \frac{n(n-1)}{2} = \frac{n^2+n}{2} = \binom{n-1}{2}$.

The number of vertices in our data structure represents the total number of mined blocks. The number of edges in our data structure is lower than the maximum number of edges in a DAG. As every arborescence is also a DAG (but not every DAG is an arborescence), we can use the same algorithm with the advantage that each block, except for the genesis, has exactly one outgoing edge to its predecessor s.t. $O(|V| + |V - 1|)$.

If two or more chains are equally long, one of the equally long chains is selected deterministically and returned by the function $\text{MAIN}(\cdot)$. Hereby it is not important how

tie-breaking works in detail, only that they are deterministically resolved. For example, by selecting the chain where the last block has the smaller hash value².

Definition 23 ($\text{MAIN}(InTree)$). We define the function $\text{MAIN}(InTree)$, which returns the main chain, i.e., the longest path in a given directed acyclic graph ($InTree$), or if there is a tie, one of the equally long chains has to be selected deterministically.

Equation 7.9 shows an example application of the main chain finding function in our notation:

$$c_3 := (b_1, b_2, b_2) \quad (7.5)$$

$$c_4 := (b_1, b_1, b_2, b_3) \quad (7.6)$$

$$c'_3 := (b_0, b_1, b'_2) \quad (7.7)$$

$$InTree := \{c_3, c_4, c'_3\} \quad (7.8)$$

$$\text{MAIN}(InTree) = c_4 \quad (7.9)$$

The fundamental mechanism by which Bitcoin and similar PoW-based *permissionless* cryptocurrency systems reach agreement depends upon consensus participants extending a proof-of-work (PoW) weighted hash chain, i.e., a *blockchain* (c). Specifically, it is assumed that a sufficient honest majority of these participants, so-called *miners*, will only build upon the branch with the most cumulative PoW, where each element, i.e., *block* (b), adheres to some pre-agreed set of protocol rules Π under which it is considered valid. As the validity of a block b can only be assessed in the context of its predecessors, we use (c_n, b_{n+1}) as an abbreviation for appending a block to a given chain, i.e., $(b_1, b_2, \dots, b_n, b_{n+1})$. This results in a sequence of blocks (i.e., a blockchain), where b_{n+1} is the latest block in this specific chain, also called the head of this specific chain.

Relation of Transaction and Blocks

Any new block which should be proposed has to be based on the set of available transactions. Hereby, we differentiate between *well-formed* and *valid* transactions. In prevalent cryptocurrencies, the validity of a transaction (as well as the validity of the associated cryptographic signatures) can only be assessed given all its predecessors, i.e., a specific chain of blocks leading to this transaction. Thus, validity can only be determined in the context of a specific history of state transitions, i.e., previous blocks. In contrast, a transaction can be well-formed if there exists the possibility for the transaction to be valid in a particular state. Thereby, it is irrelevant if this required state is ever reached in a specific instance of the protocol. An example would be a correctly signed transaction that complies with the formatting rules of the protocol, but the underlying addresses do not have any associated funds yet.

²Naïve tie-breaking methods can lead to security concerns as they might provide an advantage for an attacker (e.g., [ES14]), so special care has to be taken in practice.

Definition 24 ($\text{WELLFORMED}(tx)$). The predicate WELLFORMED returns TRUE if the given transaction can be valid in the context of at least one chain of valid blocks and their transactions. There exists a block in the set of all possible valid blocks, which includes the respective transaction.

$$\text{WELLFORMED}(tx) := \begin{cases} \exists b \in \mathcal{O}, \text{VALID}(b) \wedge tx \in \text{TXSEQ}(b) & \text{TRUE} \\ \text{otherwise} & \text{FALSE} \end{cases} \quad (7.10)$$

Depending on the set of available well-formed transactions, certain future blocks become theoretically reachable or stay unreachable since the respective valid operation to reach it cannot be created. To reason about this aspect of permissionless blockchains systems, we define a predicate REACHABLE , which returns true if a desired block is reachable given a starting block and a set of not already included well-formed transactions.

Definition 25 ($\text{REACHABLE}(b_n, b_m, \mathcal{T})$). Returns TRUE if block b_m is reachable starting from block b_n using only blocks containing well-formed transactions provided in \mathcal{T} , which have not yet been included in the history of b_n .

$$\text{REACHABLE}(b_n, b_m, \mathcal{T},) := \begin{cases} b_n = b_m & \text{TRUE} \\ b_n \neq b_m & \text{FALSE} \\ \exists tx \in \mathcal{T}, tx \in \text{TXSEQ}(\text{CHAIN}(b_n)) \wedge m > n & \text{FALSE} \\ \text{REACHABLE}(b_n, b_{m-1}, \mathcal{T} \setminus \text{TXSEQ}(b_m)) \wedge \text{TXSEQ}(b_m) \in \mathcal{T} & \text{TRUE} \\ \text{otherwise} & \text{FALSE} \end{cases} \quad (7.11)$$

$$\begin{cases} b_n = b_m & \text{TRUE} \\ b_n \neq b_m & \text{FALSE} \\ \exists tx \in \mathcal{T}, tx \in \text{TXSEQ}(\text{CHAIN}(b_n)) \wedge m > n & \text{FALSE} \\ \text{REACHABLE}(b_n, b_{m-1}, \mathcal{T} \setminus \text{TXSEQ}(b_m)) \wedge \text{TXSEQ}(b_m) \in \mathcal{T} & \text{TRUE} \\ \text{otherwise} & \text{FALSE} \end{cases} \quad (7.12)$$

Based on our definition of the predicate $\text{REACHABLE}(\cdot)$ we define a function $\text{FUTURE}(\cdot)$ returning the set of possible future blocks given a starting block and a set of available well-formed transactions.

Definition 26 ($\text{FUTURE}(b_n, \mathcal{T})$). Returns the set of reachable, i.e., constructible, future blocks $\vec{\mathcal{O}}$ given a starting block and a set of available well-formed transactions that have not yet been included in a chain determined by the block b_n .

$$\text{FUTURE}(b_n, \mathcal{T}) := \{b \in \mathcal{O} \mid \text{REACHABLE}(b_n, b, \mathcal{T})\} \quad (7.13)$$

If $b_n = b_1$ and all potentially constructible well-formed transactions are available in \mathcal{T} , then $\vec{\mathcal{O}} = \mathcal{O}$. In any case, it holds that $\vec{\mathcal{O}} \subseteq \mathcal{O}$.

A blockchain can thus also be viewed as a sequence of blocks (i.e., operations according to SMR), which reduces the cardinality of the set of all reachable valid future blocks. The reason for this is that any particular block negates the validity of all potential other blocks at the same height. Thereby, a block influences the validity of potential future

transactions and, thus, also potential future blocks. In turn, this influences the set of all potentially reachable future system states.

The function `FUTURE` returns the set of reachable operations $\vec{\mathcal{O}}$ given a starting state and a set of available well-formed transactions. First, we provide an example where all (theoretically constructible) well-formed transactions are assumed to be available in \mathcal{T} . As the total number of reachable blocks decreases with every block for any two consecutive blocks, it holds that $\forall b_n \in \mathcal{O} \ |\text{FUTURE}(b_{n-1}, \mathcal{T})| > |\text{FUTURE}(b_n, \mathcal{T} \setminus \text{TXSEQ}(b_{n-1}))|$. Any progress in the blockchain can thus be viewed as shrinking the overall set of reachable future blocks (and therefore also states) by settling on a specific history.

We have now defined all required components to construct a Nakamoto consensus protocol.

7.1.1 A Nakamoto Consensus Protocol

Using our previous definitions in combination with some assumptions regarding the network, we can now define and instantiate a Nakamoto consensus (NC) protocol. Thereby, we rely on the properties *persistence* and *liveness* for robust transaction ledgers, which were first described in [GKL20]. Since then, variations thereof have been used in various works such as [DKT⁺20, GKR20]. The details regarding the definition of these two desired properties slightly differ within the respective papers. Therefore, we just provide an intuitive description sufficient to understand the essence of those properties across different papers and formalisms.

- **Persistence** (referred to as consistency in [GKR20]) parameterized by k , referring to the number of suffix blocks to discard s.t. all honest nodes are synchronized and agree on the remaining blocks of their longest chain.
- **Liveness** is parameterized by u , referring to the number of slots in which there is at least one block contributed from an honest party in his longest chain.

In a secure NC-like longest chain protocol, these properties should hold with high probability in relation to their security parameters. In general, the security of NC-like longest chain protocols has been shown as long as honest nodes contribute proportionally more blocks (i.e., are able to produce more proof-of-work successes) than adversarial nodes. According to [GKR20, DKT⁺20], the concrete ratio is dependent on the network delay and the respective success rates s.t.

$$r_a < \frac{1}{\Delta_0 + 1/r_h}, \quad (7.14)$$

where r_h is the expected number of honest PoW successes per time unit, and r_a is the expected number of adversarial successes. In addition, there exists the assumption that no message is delayed more than Δ_0 time units. Hereby, the respective PoW successes are modeled as independent Poisson processes. More accurately, a discrete approximation to the Poisson distribution is used in [GKR20], in which time is divided into small slots

of length s and where the probability of an adversarial success in a single slot is given by $p_a = s \cdot r_a$ (and vice versa for the honest case). The network delay, in terms of the maximum number of slots an adversary can delay a message, is given by $\Delta = \lceil \Delta_0/s \rceil$. Under this model and assumptions, a NC protocol fulfills the desired properties of *persistence* and *liveness* if,

$$p_a < \frac{1}{\Delta - 1 + 1/p_h}. \quad (7.15)$$

For the simple case, where $s = 1$ and there is no network delay, s.t., every message is available in the next round $\Delta = 1$. This gives us the common majority bound on the adversarial hashrate:

$$p_a < \frac{1}{1/p_h} \quad (7.16)$$

$$p_a < p_h \quad (7.17)$$

As we are primarily interested in the characteristics of NC when interpreted as a game, including economically rational actors, we omit further details and provide an instantiation of a concrete NC protocol under our model. The proofs in [GKR20, DKT⁺20, GKL20] are constructed in a very generic way s.t. they can be applied to different NC-like protocols. Therefore, if only honest/altruistic and adversarial/Byzantine actors are considered, also our instantiation is covered. For the details regarding the associated security proofs for NC-like protocols in a setting where there are only honest players and adversarial ones, we refer the reader to the respective papers [GKR20, DKT⁺20, GKL20].

Definition 27 (NC Protocol). We define a NC protocol Π as 5-tuple:

$\Pi := \langle b_1, \text{EXECUTE}(\cdot, \cdot), \text{VALID}(\cdot), \text{PROPOSE}(\cdot, \cdot), \text{MAIN}(\cdot) \rangle$, where b_1 is the genesis block, which defines the starting state s_1 , i.e., $(s_1, rst_1) := \text{EXECUTE}(\perp, b_1)$. Under the assumption that participants exchange information on blocks and transactions over direct point-to-point communication channels in a Δ -synchronous network setting, a NC protocol Π maintains a *robust public transaction ledger* [GKL20, DKT⁺20] if the ledger is organized as blocks of transactions and satisfies the properties *persistence* and *liveness*.

In comparison to the *backbone model* [GKL20], our predicate $\text{VALID}(\cdot)$ is comparable to the `validate` algorithm parameterized by the *content validation predicate* $V(\cdot)$. The $\text{MAIN}(\cdot)$ function is comparable to the `maxvalid` functionality used for chain comparison, and the PROPOSE function resembles the `pow` algorithm in the backbone model. The $\text{EXECUTE}(\cdot)$ function is comparable to the *chain reading function* $R(\cdot)$. The *input contribution function* $I(\cdot)$ is the subject of Section 7.1 and onwards, where we discuss how different types of economically rational players would interact with the protocol and the resulting challenges for those players as well as for the protocol as a whole.

7.1.2 Finite Blocks and Finite Chains in Finite Instances

In the context of most NC-based cryptocurrencies, a state is represented by a sequence of n blocks starting from the genesis block, e.g., $s_n = (b_1, b_2, \dots, b_n)$. In Bitcoin, for example, an operation is equivalent to a block b , which itself contains a finite ordered sequence of transactions. As in our model, blocks have a size limit. This implies that the number of transactions in a block, as well as their size, is also bounded and thus finite. By our definition, each transaction has size $1/\aleph$. Thus we can denote the maximum number of transactions in a block with \aleph . This is a difference compared to other models, such as [GKL15, GKL20], where blocks are unbounded in size to ensure that transactions cannot be efficiently censored to achieve *liveness*. Instead, in [GKL15, GKL20], the number of messages (including transactions) per round is bounded by the used computational model of ITM (interactive Turing machines) from [Can00].

Note that we also implicitly restrict the set of potentially valid operations \mathcal{O} to be finite by requiring that the predicate $\text{VALID}(\cdot)$ has to return a value for all operations. This requires checking the validity of all previous operations in the respective chain. As the predicate $\text{VALID}(\cdot)$ is required to be computable, i.e., terminate on all inputs, this prohibits infinitely long chains from being valid.

In the case of permissionless PoW, this restriction to finite blockchains leads to an interesting observation. As pointed out by Pass et al. [PS17], PoW blockchains cannot stop without becoming insecure, as equally long or even longer heavier chains can be constructed with a different history after the protocol has terminated. This would make it impossible for a new participant to distinguish a chain created in retrospect from the original discontinued longest chain. Thus, permissionless PoW blockchains are required to run infinitely long in theory³. To not contradict this requirement, we restrict our model to *finite instances of protocol executions*. This, in turn, can only lead to finite sequences of operations, i.e., blocks describing state transitions, which can be used as input to the validity predicate and return true if valid.

7.2 Types of Protocol Changes and Parallel Executions

We now use our model to describe, classify and compare different types of forking events and relate them to the relevant literature. Thereby, we want to demonstrate the usefulness of our model even for corner cases in NC-like protocols and highlight which conditions make the difference between a so-called *soft-fork*, where a *chain split* typically does not occur, and a so-called *hard-fork* in which a permanent chain split leads to the existence of two parallel systems. We divide these *permanent* forking events into *protocol changes* and *parallel executions*.

³In relation to this, another interesting observation regarding the necessity of protocol updates is outlined in Appendix A.9.

The concept of *temporary* forks is a core design property of Nakamoto consensus (NC). Even without the interference of malicious actors, the proposal of a block by two different miners at approximately the same point in time can trigger a fork. Such a temporary fork eventually resolves as soon as new blocks on top of either branch are found and propagated. The non-deterministic nature of the hash-based PoW employed in such systems, as well as the relatively weak synchrony assumptions of the underlying peer-to-peer network, can lead to situations where multiple valid branches are created and extended in parallel. However, the probability of such a blockchain fork prevailing for prolonged periods decreases exponentially in its length if a sufficient majority of miners only extend the heaviest chain (known to them) and the synchrony assumptions for the network still hold [Nak08, GKL15]. This brings us to the question of how a change $\Pi \rightarrow \Pi'$ to the underlying protocol rules may affect this consensus mechanism.

Apart from these temporary forks that are resolved when one branch takes the lead, there also exists the concept of a *permanent fork*, also referred to as a *chain split*. In such a case, the end result is a split into two different systems. This can happen intentionally or unintentionally, for example, when the underlying protocol has to be updated. In this context, the loosely defined terms *hard-fork*, *soft-fork*, and *velvet-fork* have established themselves as descriptors of different classes of upgrade mechanisms for the underlying rules. The term *hard-fork* has established itself [BMC⁺15, GCR16] as a descriptor for protocol changes that can incur a permanent split of the blockchain, as they permit or even enforce the creation of blocks considered invalid under previous protocol rules. In contrast to this, *soft-forks* intend to retain some level of compatibility with older protocol versions, specifically for clients adhering to previous protocol rules [Bitc, Bitb]. In contrast to temporary forks, which can happen during normal operation, hard- and soft-forks refer to a protocol upgrade that might or might not lead to a permanent (scheduled) fork, i.e., a chain split. Therefore, we will refer to them more accurately as *protocol changes* (e.g., in the form of updates) in the following sections.

In the following, we will investigate different situations of disagreement on the validity of a block b under different rules, i.e., protocol changes. If there are both types of nodes in the network following either definition of rules, this can lead to a permanent chain split in the blockchain, where a subset of participants will always reject branches building on a , to them invalid, block, regardless of the cumulative PoW these branches accumulate. We now present an overview and definitions of these different upgrade mechanisms and outline their relationships. Hereby, we expose examples where different types of modification techniques or related constructions have already been used in cryptocurrencies. Furthermore, we expand upon the concept of different protocol changes by discussing potential security implications that can arise from different approaches if used as part of an attack.

7.2.1 Types of Protocol Changes and Parallel Executions in the Literature

The term *hard-fork* has established itself [BMC⁺15, GCR16] as a descriptor for protocol changes that can incur a permanent split of the blockchain, as they permit or even enforce the creation of blocks considered invalid under previous protocol rules. As an alternative, *soft-forks* intend to retain some level of compatibility with older protocol versions, specifically for *clients* adhering to previous protocol rules. The concepts of both hard- and soft-forks are described in the Bitcoin developer guide [Bita], as well as the Bitcoin-Wiki [Bitb]. In scientific literature, some of the principal differences between these two types of consensus rule upgrades have been covered in [BMC⁺15, GCR16, CDFZ17]. McCorry et al. furthermore provides a history of forking events in both Bitcoin and Ethereum as part of their work on how parties can bindingly perform atomic cross-chain trades in case of a permanent blockchain fork [MHM17]. A closer description of different protocol forking mechanisms and their relation to each other was also presented in a blog post by Buterin in [But17], where Buterin distinguishes between *strictly expanding* and *bilateral* hard-forks. Bilateral hard-forks can prevent protocol interactions between permanent forks by rendering their rules incompatible, which may not be ensured in the case of a strictly expanding hard-fork. As an example, consider mutually valid transaction rules, which can render transactions valid on either blockchain of a permanent fork and can lead to replay attacks. Apart from all these mentions of hard- and soft-forks and high-level descriptions, the terms have never been properly defined, although they are being used throughout the cryptocurrency community.

If we consider the possibility of a permanent blockchain split as the defining characteristic of hard-forks, most protocol changes would fall into this category. For example, reducing the validity set of rules in a protocol update, which is generally considered to be a *soft-fork*, can lead to a permanent split in case the majority of consensus participants are not upgraded. Conversely, if an expanding protocol change, i.e., a *hard-fork*, does not reach a majority among consensus participants, no permanent fork is actually incurred as upgraded clients will continue to follow the chain with the most cumulative PoW (for more details see the examples provided in table 7.1). By this example, it becomes apparent that the distinction between hard- and soft-forks is not as clear. This dichotomy helps outline the difficulties in presenting a clear distinction between the different types of forks.

7.2.2 Types of Protocol Changes

To provide a finer distinction between the possible impacts of protocol upgrades in NC and their potential for permanent chain splits, we first present different classes of protocol changes (see Table 7.1) from old protocol Π to a new protocol Π' with respect to their influence on the set of valid blocks (operations) \mathcal{O} , i.e., possible blockchains, which changes to \mathcal{O}' under the new rules. Depending on the degree of support a rule change receives amongst participating nodes, referred to as *miners*, the system might split into two systems following either set of protocol rules (Π and Π' respectively) or

remain united under the set of rules followed by the majority of miners. There might also be changes or designs of protocols that are constructed to be incompatible with another system right from the start, which immediately results in the parallel execution of two different systems (for such a construction, we refer to Section 7.2.3).

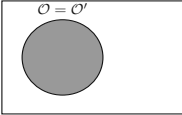
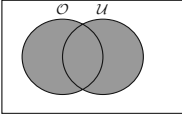
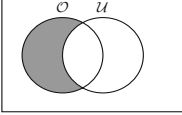
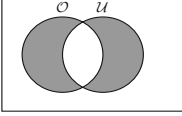
Type	Set of valid blockchains:		Incurred Split:		Examples
	Venn Diagram (\mathcal{O}' is gray)	Relation	\mathcal{O} is majority	\mathcal{O}' is majority	
No change		$\mathcal{O}' = \mathcal{O}$ $\mathcal{S}' = \mathcal{S}$	-	-	Normal operation
Expanding Operations		$\mathcal{O}' = \mathcal{O} \cup \mathcal{U}$ $\mathcal{O}' \supset \mathcal{O}$	no chain split (<i>soft-fork</i>)	chain split (<i>hard-fork</i>) triggered when: $b \in \mathcal{U} \setminus \mathcal{O}$	Blocksize increase, new opcode
Reducing Operations		$\mathcal{O}' = \mathcal{O} \setminus \mathcal{U}$ $\mathcal{O}' \subset \mathcal{O}$	chain split (<i>hard-fork</i>) triggered when: $b \in \mathcal{O} \cap \mathcal{U}$	no chain split (<i>soft-fork</i>)	Blocksize decrease, opcode removal, SegWit
Overlapping and Conflicting Operations (<i>Bilateral</i>)		$\mathcal{O}' = (\mathcal{O} \cup \mathcal{U}) \setminus (\mathcal{O} \cap \mathcal{U})$ $\mathcal{O}' = \mathcal{O} \Delta \mathcal{U}$ $\mathcal{O}' \cap \mathcal{O} \neq \emptyset$	chain split (<i>hard-fork</i>) triggered when: $b \in \mathcal{U}$ no chain split (<i>soft-fork</i>) when $b \in \mathcal{O} \setminus \mathcal{U}$	chain split (<i>hard-fork</i>) triggered when: $b \in \mathcal{U}$ no chain split (<i>soft-fork</i>) when $b \in \mathcal{O} \setminus \mathcal{U}$	Opcode redefinition, chain ID for replay protection in ETH/ETC fork

Table 7.1: Overview of classes of protocol changes from old to new rules $\Pi \rightarrow \Pi'$. Thereby, \mathcal{O} and \mathcal{O}' denote the sets of valid blocks under the old and new protocol rules. The new set of valid blocks \mathcal{O}' is colored gray. \mathcal{U} denotes the validity set changes introduced by the protocol update.

No Change

In the simplest case, there is no rule change and hence no new set of valid blocks \mathcal{O}' . In this scenario, all attacks and temporary forks can be described as changes in the longest block sequence, and thus lie within \mathcal{O} . The attractiveness to engage in malicious temporary forks depend on the profitability and reachability of certain blocks (see Figure 7.1).

Expanding Operations

The new protocol rules Π' increase the set of blocks considered valid with respect to the previous protocol rules Π , i.e., $\mathcal{O}' \supset \mathcal{O}$. Hereby, the update with the set of new blocks that become valid under the new rules Π' is denoted \mathcal{U} . Expanding protocol changes can cause a permanent split in the blockchain if the consensus participants adhering to Π' form a majority. However, if a majority retains protocol rules Π , no permanent chain

split occurs, as clients adhering to Π' also consider any blockchain under protocol rules Π as valid. Examples include blocksize increase or defining previously unused values as new Bitcoin/Ethereum opcodes.

Reducing Operations

The new protocol rules Π' reduce the set of blocks considered valid with respect to the previous protocol rules Π . Specifically, the new set of valid blocks \mathcal{O}' is a proper subset of the valid blocks of the previous protocol, i.e., $\mathcal{O}' \subset \mathcal{O}$. Reducing protocol changes do not lead to a permanent chain split, i.e., they represent a soft-fork, as long as the majority of consensus participants adhere to the new rules Π' . If, however, Π retains a majority, a permanent chain split is incurred as updated clients and miners will consider some blocks that are valid under old protocol rules Π as invalid. Examples could be a blocksize decrease, the introduction of SegWit (BIP 141 [LLW]) and the removal of an opcode.

Overlapping and Conflicting Operations (Bilateral)

We refer to updates introducing mutual incompatibilities as *conflicting* or *bilateral* protocol changes. Here, the goal is to intentionally cause a permanent fork of the blockchain and prevent potential interactions between the resulting chains, such as the chain ID introduced in Ethereum for replay protection or redefinition of a Bitcoin opcode. Although both systems share a common history, they also have *overlapping* operations considered valid in both systems. Therefore, as long as overlapping blocks in the common history $\mathcal{O} \setminus \mathcal{U}$ are mined by the majority of miners, no permanent chain split will be triggered. This would resemble the situation before a scheduled block height, after which the rule change is activated.

If a permanent chain split is triggered, this immediately results in a situation where there are multiple resources that have to be considered when calculating the EEV, i.e., the old and the new cryptocurrency in which a participant could have assets. This directly leads to the question of optimizing protocol compositions.

7.2.3 Types of Parallel Executions

Some of the protocol changes described in Section 7.2 can lead to permanent chain splits s.t. in the end there exist two different systems in parallel. As such systems might share some common history, or even certain rules, this can lead to interesting interactions between systems running in parallel. An example for a problematic interaction, would be the missing replay protection for transactions when Ethereum Classic split up from Ethereum⁴.

This section describes different types of interactions in parallel protocol executions (i.e., compositions) and how they can be used or have been used in different AIM attacks.

⁴cf. <https://eips.ethereum.org/EIPS/eip-155>

Therefore, we need to look at scenarios in which there exist at least two systems in parallel to investigate how these systems can influence each other. This inherently requires that there exist at least two systems players care about and that they participate in or actively follow events in the respective systems. By cryptographically interlinking actions in one system with actions in another cryptocurrency, the two systems can become interleaved. Thus, the incentive to perform a certain action is not only tied to in-band profit opportunities but also to out-of-band profit opportunities. Using such methods, also incentives for temporary forks or permanent protocol changes can be created in other systems. Examples of such constructions have been described in [JSSW18, JSZ⁺21a, JSZ⁺21b, Ros21], as well as in Chapters 2, 3, and 4 of this thesis. Table 7.2 shows the different types of parallel protocol executions and their potential for interference.

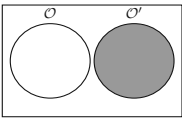
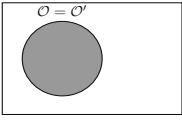
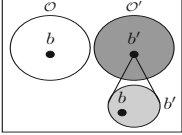
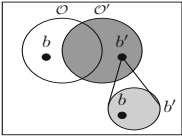
Type	Set of valid blocks:		Description of Different Protocols:		Examples
	Venn Diagram (\mathcal{O}' is gray)	Relation	Π	Π'	
Disjoint Operations		$\mathcal{O}' \cap \mathcal{O} = \emptyset$	org. protocol (e.g., Bitcoin)	new protocol (e.g., Litecoin)	Creation of new cryptocurrencies like Litecoin etc.
Alternative Interpretation of Operations (<i>Velvet-fork</i>)		$\mathcal{O} = \mathcal{O}'$ $\mathcal{S} \neq \mathcal{S}'$	no chain split in org. protocol unaware users may produce only states in \mathcal{S}	no chain split in new protocol aware users can produce states in \mathcal{S}'	Colored coins [Ros12], Adoption of NIPoPoWs [KMZ17]
Superset Operations		$\mathcal{O}' \cap \mathcal{O} = \emptyset$ $\exists b \in \mathcal{O} \exists b' \in \mathcal{O}', b \subset b'$	org. protocol	new protocol, some blocks are supersets of blocks from org. protocol	Merged mining [JZS ⁺ 17], P2Pool [P2P], Out-of-band variants of pay-to-win Attack [JSZ ⁺ 19], GoldfingerCon [MHM18]
Overlapping and Superset Operations		$\mathcal{O}' \cap \mathcal{O} \neq \emptyset$ $\exists b \in \mathcal{O} \exists b' \in \mathcal{O}', b \subset b'$	org. protocol	new protocol shares some common history with org. protocol, new blocks might incorporate blocks of org. protocol	Pitchfork [JSSW18]

Table 7.2: Overview of different types of protocol compositions of two protocols Π (original) and Π' (new). Thereby, \mathcal{O} and \mathcal{O}' denote the sets of valid blocks under the different protocol rules.

Disjoint Operations

In the simplest case, parallel systems have disjoint sets of valid operations, i.e., have no overlap with each other and thus coexist without any direct interference. This is the case for any newly created cryptocurrency with its own consensus algorithm and genesis block, for example, Bitcoin and Litecoin. Both are cryptocurrencies based on NC with PoW, but they use different PoW algorithms and different genesis blocks. Therefore,

those two protocols do not interfere directly with each other, e.g., actions in one protocol cannot be replayed in the other protocol.

Due to the characteristics and usage of cryptography in prevalent cryptocurrencies, even systems with disjoint sets of valid operations can be connected by creating a third system that combines operations from both systems within it. A payment channel would be one example. For more information, see the Section on *superset operations* below.

Alternative Interpretation of Operations (Velvet-fork)

A so-called *velvet-fork* [KMZ17] seeks to add additional meaning to operations without affecting the validity rules of the original protocol. More accurately, a velvet-fork creates a new protocol Π' , which provides an alternative interpretation of inputs to some original protocol Π . The new protocol rules Π' only change the execute function s.t. it invokes and computes the original execute function with the current state of the original protocol s_n and the new block b_{n+1} as input. This is required to deduce if a new valid state is computed in the original protocol, i.e., the result is not \perp . If so, the new execute function $\text{EXECUTE}'$ is invoked with the current state of the new protocol s'_n and the exact same block b_{n+1} from the original protocol as input. This function then potentially produces a different output state s'_{n+1} and result (which is not \perp) for the new protocol Π' . Therefore, the set of potentially reachable systems states in Π' changes ($\mathcal{S}' \neq \mathcal{S}$), but the overall set of valid operations stays the same $\mathcal{O}' = \mathcal{O}$.

The two protocols, Π and Π' , are thus executed in parallel, and every input to Π is also forwarded to Π' . So only if blocks or transactions are *valid* under the old (and thus also the new rules) the new rules can be applied in addition and potentially lead to a new state in the new protocol Π' . The resulting state can differ from the state of the original protocol. If a block does not produce a result, the new rules are ignored as well, as previous protocol rules Π are relied upon to determine validity.

Since these changes only provide an alternative interpretation in a newly created system, velvet-forks never incur a (permanent) protocol fork or chain split, as any element considered valid under Π' is also considered valid under Π , therefore $\mathcal{O}' = \mathcal{O}$. The term *velvet-fork* can hence be considered somewhat of a misnomer. Examples of such systems are colored coins [Ros12] or the adoption of NIPoPoWs [KMZ17].

Colored Coins: A concept closely related to the idea of velvet-forks is that of overlay protocols and colored coins, inter alia described in [Ros12]. The term colored coin refers to cryptocurrency transactions where the outputs are additionally “colored“ to represent some assets or tokens, allowing to use of such outputs in transactions to transfer their ownership. Rosenfeld presents an overview of colored coins and describes a possible implementation approach in [Ros12]. We consider colored coins to be part of the class of overlay protocols and herein focus on the latter, more general concept. Overlay protocols leverage an underlying property of NC systems, namely, providing an ordering of transactions. Under the assumption that NC eventually produces a total ordering

of transactions within its common prefix that is unlikely to change, the result can be compared to *total order broadcast* or *atomic broadcast*, which has been shown to be equivalent to consensus [CT96] and can, for instance, be used to readily implement state machine replication. As such, encoding messages in regular valid transactions allows overlay protocols to utilize NC or similar systems as if they were a “eventual total order broadcast” protocol. While this approach may provide overlay protocols with a mechanism for eventually reaching agreement on the ordering of messages, it does not extend any guarantees towards their *correctness*. In particular, miners may remain completely oblivious to the rules Π' of the overlay protocol and only adhere to the underlying base protocol Π . Hence, transactions encoding invalid messages of the overlay protocol Π' , which have to be ignored by the participants of the overlay system, may be included in blocks [BMC⁺15]. However, if participants in the overlay system agree to both the same set of rules Π' and the “eventual ordering” of both valid and invalid messages, then ignoring messages considered invalid under Π' by all (honest) participants leads to the same (eventually) consistent system state. Overlay protocols are a form of velvet-fork in that they impose no restrictions and apply new protocol rules Π' only if the input is considered valid.

Superset Operations

In this case, a block in \mathcal{O}' can be a superset of operations in \mathcal{O} containing one or multiple operations from \mathcal{O} . In other words, a block $b' \in \mathcal{O}'$ can contain a block $b \in \mathcal{O}$ s.t. $b \subset b'$. Because the set \mathcal{O}' is not transitive, no block from \mathcal{O} is directly valid as a block in \mathcal{O}' , hence $\mathcal{O}' \cap \mathcal{O} = \emptyset$. Examples of such parallel construction are merged mining [JZS⁺17], P2Pool [P2P], and out-of-band variants of pay-to-win attacks [JSZ⁺19], as well as GoldfingerCon [MHM18].

Under the assumption that there are overlapping sets of participants, using such constructions, it is possible to create interdependencies between operations in two systems with disjoint operations. The ephemeral chain relay constructed in [JSZ⁺19], as well as in Chapter 4, is an example of such a causal dependency. Other examples of compositions of protocols with superset operations are P2Pool and merged mining.

P2Pool: P2Pool [P2P] is a protocol for implementing decentralized mining pools presented in 2011. In contrast to conventional mining pools, attestation of each miner’s contribution to solving the next block’s PoW puzzle and the distribution of rewards are accomplished without a trusted operator. P2Pool uses an additional, length-bounded blockchain, the *sharechain*, consisting of otherwise valid blocks which fail to meet the mining difficulty target d but exceed a minimal target d_{share} , agreed upon and determined by the protocol⁵, sometimes referred to as *near* or *weak blocks*. These blocks are used to attest each miner’s contribution, while the reward distribution is, in turn, achieved by introducing the following rule: “Each time a miner finds a block exceeding the

⁵The target d_{share} is adjusted such that the sharechain maintains an average block interval of 30 seconds.

target d , she can claim 0.5% of the block reward, while the rest must be distributed among all participating miners according to their portion of the last N sharechain blocks". This additional rule is transparent to the mined blockchain, i.e., it generates fully backward compatible blocks and hence remains oblivious to all but P2Pool miners, which additionally follow the sharechain, i.e., the protocol executing in parallel. As a result, any valid block generated by P2Pool miners adhering to the rules of Π' will be accepted by non-P2Pool miners in Π . In turn, P2Pool miners only accept blocks in the sharechain if they follow the agreed reward distribution. Moreover, as the sharechain accepts blocks with lower difficulty, not all sharechain blocks are valid in Π .

Merged Mining: Merged mining refers to the process of reusing (partial) PoW solutions from a *parent* blockchain as valid proofs-of-work for one or more *child* blockchains [JZS⁺17]. It was first introduced in Namecoin [Namb] both as a bootstrapping technique and to mitigate the fragmentation of computational power among competing cryptocurrencies sharing the same PoW. While a *child* cryptocurrency is required to implement and support merged mining, *parent* blockchains only need to allow miners to include additional arbitrary data in their blocks. This arbitrary field is then used to link blocks to the merge mined child cryptocurrency. From the perspective of the parent cryptocurrency, merged mining can be considered a velvet-fork, as new consensus rules, namely those of the merge mined children, are incorporated in the parent blockchain in a fully backward compatible way. If either invalid or no links to child blocks are included in a block, the data will be ignored by participants of merged mining, but the block is nevertheless accepted in the parent chain as usual. From an overall perspective, merged mining is a composition of at least two protocols, whereby there is one protocol (the child blockchain) in which there exist operations that are a superset of operations (i.e., blocks) in another protocol called the parent blockchain. As the parent blockchain block carrying the PoW is only a subset of the entire child chain block, merged mining can be considered a perfect example of a protocol composition with superset operations. Merged mining and P2Pool make use of the same principle mechanisms, with the marked difference being that in merged mining, additional rewards are received in the child cryptocurrency, whereas P2Pool sharechain blocks represent claims to portions of the next valid block reward on the main chain.

Overlapping and Superset Operations

A combination of overlapping and superset operations is also possible. In this case, Π and Π' share a common history. In addition, operations in \mathcal{O}' might contain valid blocks from \mathcal{O} after a permanent chain split has occurred. An example would be the Pitchfork attack [JSSW18]. In this attack, a chain split is scheduled, and after this event, every new block in \mathcal{O}' is required to link to a valid block in \mathcal{O} . Therefore, it holds that $\exists b' \in \mathcal{O}' \setminus \mathcal{O} \exists b \in \mathcal{O} \setminus \mathcal{O}', b \subset b'$.

Due to the chaining property of blockchains, overlapping blocks only exist if both systems share a common history, i.e., genesis block. Therefore, it is not directly possible for a

single block of two different instances to overlap if they do not share a common ancestor block. This is a difference compared to a paper by Lindell et al. [LLR04], in which the authors prove that authenticated Byzantine Agreement protocols only remain secure under parallel or concurrent composition (even for just two executions) when more than $2/3$ of the participating parties are honest. The reason for this is that participants do not know in which execution they currently are. The solution to the problem is the usage of unique identifiers or sequence numbers for each execution. Blockchains inherently have this property due to the chaining of blocks

In comparison, superset operations can always be used when the state of a targeted NC system is required as input in another system. The ability to interconnect different NC-style systems and thereby influence the economic incentive structures of those systems is further explored in Sections 7.5, 7.6, and 7.7. In these sections, we describe and use the concept of *adversarial games* (AG) to construct a new system Π' , which decides, depending on the outcome of the original system Π , whether a preferable state was reached and then issues rewards accordingly. Depending on the capabilities of the targeted system, the promised rewards can be issued using in-band payments or out-of-band payments. The three adversarial games should illustrate how to use our model to construct and describe such attacks with varying capabilities and interference with consensus and what the necessary assumptions for those attacks are.

7.3 Challenges of Modelling Nakamoto Consensus-based Cryptocurrencies as Games

So far we have only discussed NC without considering incentives. Most cryptocurrencies also provide some incentive to participate by issuing some form of *reward* for advancing the protocol. Due to this rewarding mechanism, such cryptocurrencies can also be viewed as games. In this section, we want to describe some basic characteristics of NC-based cryptocurrencies when considering incentives and then highlight challenges in modeling NC-based cryptocurrencies as games according to game theory.

In Section 7.4 we describe a simplified *blackbox approach* to abstract all game elements and the concept of incentives by a function that determines the expected extractable value (EEV), or in other words, the *utility* of a block for a player. For a more detailed definition and discussion of EEV we refer to [JSSW22a] or Chapter 6 of this thesis .

7.3.1 Game like Characteristics of NC-based Cryptocurrencies

In this section, we augment the already described NC protocol and describe a basic *reward mechanism* that captures the essential characteristics of the rewarding mechanisms

found in prevalent cryptocurrencies based on NC. The result is a Nakamoto Consensus game (abbreviated as NC game). This game shares some resemblance with infinitely repeated stochastic games in game theory and was already modeled as such in previous modeling attempts [KKKT16]. In practice, any NC protocol is executed by a set of *participants*. Therefore, the NC game has to be played by a set of *players* \mathcal{P} . The total number of players is denoted by $|\mathcal{P}|$. Each participant has a certain probability p to succeed in creating a new block $\{p_1, p_2, \dots, p_{|\mathcal{P}|}\}$, where $\sum_{i=1}^{|\mathcal{P}|} p_i = 1$. In other words, for PoW cryptocurrencies p_i can be viewed as the hashrate of participant i , which depends on his computational power.

In contrast to [KKKT16], there is also a distinction between *miners* and *users* in practice. A user can be modeled as a participant who has zero probability of mining a new block⁶. Therefore, the set of miners can be defined as \mathcal{M} , where $\mathcal{M} \subseteq \mathcal{P}$, where $\sum_{i=0}^{|\mathcal{M}|} p_i = 1$. Additionally, there is a potentially empty set of users $\mathcal{U} \subset \mathcal{P}$, where $\sum_{i=0}^{|\mathcal{U}|} p_i = 0$.

In addition to a certain probability of success p_i in mining a new block, a participant i can also have funds, i.e., *resources* in terms of the associated cryptocurrency r_i . All available resources in a cryptocurrency are distributed (following some distribution) amongst participants and bounded by the overall currency limit F , s.t., $\sum_{i=0}^{|\mathcal{P}|} r_i = F$ holds at any point in a given chain of blocks, even if F changes during the runtime (height) of the chain. A transaction (tx) might change the distribution of funds amongst participants. Therefore, the current overall balance (total amount of available resources) is always tied to some block height/number n , i.e., $\forall n \in \mathbb{N}, \sum_{i=0}^{|\mathcal{P}|} r_{n_i} = F_n$.

A *state* in NC is represented by a rooted in-tree of blocks (starting at a given genesis block). In a model where miners have complete information and immediately release their blocks⁷ the NC game can be represented in extensive form by a game tree containing all possible blocks as well as the combinations in which players can make them. The outcome of a NC game is described by the blockchain that is generated from traversing the game tree in a specific path. The path to a leaf defines the order of transactions produced by the associated longest chain of blocks, which in turn have been produced by the respective miners in this path.

In prevalent cryptocurrencies, the first transaction in every block is the *payout transaction*⁸, which delivers the reward payout r_{reward} of the respective block to the address provided by its miner⁹. By assuming that this payout transaction also carries all block-

⁶As such non-mining nodes have zero chance to mine a block, their (non-zero) utility to participate comes from the fact that they can send or receive payments (see assumption 10 in Section 7.5.1).

⁷Such a model has been used in immediate-release games with the FRONTIER strategy defined in [KKKT16]

⁸This transaction is sometimes also called coinbase transaction.

⁹Note that in Ethereum, for example, there is no dedicated coinbase/payout transaction. The beneficiary is directly specified in the block header. Also, this behavior can be modeled by a *virtual* payout transaction at the beginning of the block, which specifies the reward (including fees) for the beneficiary.

specific metadata (e.g., time stamp, hash, etc.), the exact same sequence of transactions will always result in the same outcome. This is currently not the case in prevalent cryptocurrencies such as Ethereum, due to the ability of smart contracts to access block information. This has been termed semantic malleability and was outlined by Stifter et al. [SJSW22].

Apart from the payout transaction, all other transactions in a block are *payload transactions*, which are used to change the state of the system, e.g., make a *payment* to transfer funds. Therefore they are also sometimes called *payment transactions*. There is a maximum number of transactions \aleph that can be included in one block. So a block must have at least one transaction (which in this case must be the payout transaction) and at most \aleph transactions, of which $\aleph - 1$ must be payload transactions. The root block (b_1) is called the *genesis block* and cannot be changed. It includes the first payout transaction tx_0 , which might be used to initially distribute funds. So the outcome of a NC game is a sequence of blocks that might look as follows:

$$c = (b_1(tx_0), b_2(tx_1), b_3(tx_2, tx_3), b_4(tx_4), \dots, b_{|c|}(tx_{|\tau|-1}, tx_{|\tau|})) \quad (7.18)$$

The respective blocks define a total ordering of a sequence of enclosed transactions $\tau = (tx_0, tx_1, tx_2, tx_3, \dots, tx_{|\tau|})$, where $|c|$ denotes the length of the respective chain and $|\tau|$ denotes the length of the embedded sequence of transactions. This directly leads to the question, when does a chain of blocks represent a termination point in the context of a NC game? In other words, when does a (sub-)game of the NC game terminate, s.t. (some) payouts can be considered performed, or when does the entire NC game terminate? This challenge is described in

Rewards in our Model

Participating in the mining process generates costs $r_{\text{miningcosts}}$ and produces rewards r_{reward} for every block. In a basic setting, without any optimization from the miners or attacks, the reward consists of the block reward plus fees, $r_{\text{reward}} = r_{\text{blockreward}} + r_{\text{fees}}$. As the focus of this work is modeling and analyzing the feasibility of algorithmic incentive manipulation attacks, the operational mining costs play only a subordinate role, as they would occur anyway, even without any attack. For an analysis of the relationship between mining costs and attacks, we refer the reader to [TE18, CKWN16, Bud18]. Since most costs, in practice, are paid in another currency and thus require an exchange of currency units at runtime, we introduce the abstract notion of *utility* to refer to funds as well as costs in the same denomination. In our case, the value unit of account is normalized rewards ($r_{\text{reward}} = 1$) of a reference resource. We further assume that all funds can readily be exchanged into the required currency units if needed.

In contrast to other modeling attempts of NC using game theory, we focus on the feasibility of algorithmic incentive manipulation attacks under *economically rational* players. Therefore, we are primarily interested in the *execution* of the game, as well as in the *outcome*, and not particularly in finding equilibrium strategies of the original game.

7.3.2 Challenges in Modeling NC using Game Theory

In this section, we outline some challenges when modeling NC-based public permissionless cryptocurrencies using game theory.

Sub-Games and Payouts

As already stated in Section 7.1.2, a NC protocol must, in theory, run infinitely long, so there would not be any final termination point of the game. In [KKKT16], a payout for block n occurs at block $n + d$, i.e., after d blocks. These intermediate termination points also introduce a concept of finality and thereby would also work as a fork resolution mechanism for long-range forks as their length is limited to d steps. The first branch, which succeeds in having a descendant d generations later, is rewarded with a payout, and all miners will completely ignore every branch that starts at an earlier node. The general idea of the parameter d stems from a global parameter $d = 100$ used in Bitcoin, after which the payout of the mining reward (r_{reward}) is unlocked and thus spendable in a new transaction.

Although a definition of d as a fixed payout delay (such as in [KKKT16]) has proven to be useful for modeling certain game-theoretic aspects of NC, it does not accurately reflect that there is no global security parameter k in NC that *always* holds. Instead, any k only provides probabilistic guarantees, i.e., the probability that any block is reverted drops exponentially with the number of subsequent blocks (assuming network delay and the rate of honest block contributions is within the allowed bounds 7.1.1). So setting $d = k$ for receiving payouts would result in a practically workable system with parametrizable error probabilities.

Unfortunately, in practice, there is no agreed-upon global security parameter k for prevalent cryptocurrencies. Moreover, there are not only payout but also payload/payment transactions. As a result, participants choose their own values ($k_{recipient}$) for this parameter to define the waiting period after which they consider a payment transaction confirmed. This does not only have implications for payment transactions but also for payout transactions. The reason for this is that miners can use the payout transaction in their blocks to perform payments. In such a case, it is up to the recipient of the payout transaction as payment when the transaction is considered confirmed. It is possible to choose a confirmation period of $k_{recipient} < d$, even if this means the recipient cannot reuse the received funds in a new transaction till d further blocks in the future. This ties to the question: When does a (sub-)game of NCG end s.t. payouts can be considered performed?

As the game is infinite, there is no final termination point of the NC game where all payouts are performed at once. One solution approach would be to define sub-games where payouts are performed regularly, e.g., at every block x , the payout for block $x - d$ is made, where $d = k$. In other words, forks longer than d violate common prefix property and thus are unlikely in practice. Nevertheless, the probability of a fork with length $\ell > d$ is not zero. If this should ever happen, the safety of the NC protocol is violated,

and the previous state before the fork might never be reachable on a different chain due to semantic malleability [SJSW22], even if all transactions will be included in the exact same order.

In addition to block rewards (from fees and base rewards), there are also payouts received from payment transactions, i.e., all transactions which are not the first in a block (i.e., coinbase transactions), which we will also model in our approach. If we assume that every participant has chosen the same value for k , i.e., $k_{participants} = k = d$, which is the same as the parameter after which payouts are considered confirmed, then the block at position $x + d$ not only confirms the payout transaction of block x , but also all recipients of payments consider the transactions in this block as “confirmed”.

Note that it is possible that *confirmed* in the context of a block can have different meanings, depending on what it refers to. If there are different preferences of participants regarding $k_{recipient}$, then d confirms the payout transaction/reward of a block, while k_A confirms a single transaction in a block for recipient A , while k_B confirms a transaction for recipient B , but at a later point in the chain as $k_A < k_B$. This is one reason why modeling cryptocurrencies is complex. A cryptocurrency is not just *one* game, but the transactions in a cryptocurrency can be viewed as multiple games running in parallel. Moreover, all these games are not necessarily aligned with the underlying payout mechanism. The fact that there are different individually chosen $k_{recipient}$ in practice can also be used to avoid risks as a merchant. This *let's go shopping defense* is described in [JSSW22a], or Chapter 6 of this thesis

Intended Moves

In all permissionless NC-style cryptocurrencies, the concept of a binding move does not exist. Any new block b is considered an intended move and defines a non-final state transition. Moreover, there can be parallel or even contradictory intended moves that lead to temporary or even permanent forks. The difference to other classic games is that *intended moves*, compared to regular *moves* are not final. This makes cryptocurrencies with the ability to fork hard to model compared to other games.

Validity of Moves has no Time-Bound

In addition, NC-style systems allow different moves (i.e., blocks) to be issued, even later on, if a player deems this necessary. This is theoretically possible as blocks have no time bound after which they become invalid. Even old blocks can be included in their respective place in the game tree, i.e., branch/fork of the chain, as long as they result in valid state transition. If they form a new longest chain, they are considered to be the canonical chain. This means that any majority assumption regarding honest hashrate has to hold also in the future. Otherwise, past blocks might get reverted later on. In practice, the implementations of Nakamoto consensus might reject blocks belonging to a fork of a certain depth, for example, the go Ethereum client (geth), which in some

pruning modes rejects blocks longer than 128 blocks in the past. Here, the question arises, what is the theoretical foundation that justifies this exact value, and how to come up with a theory to derive such values?

Permissionlessness Exit and Join

The concept of permissionlessness is commonly understood as the ability to allow players to join and exit the (theoretically) infinite game at any point in time. Thereby, the previously infinite game becomes finite for the exiting player. This changes the properties of the game and further complicates game-theoretic modeling. Moreover, this also implicitly requires that the rate between honest and malicious players is maintained over the entire execution of the protocol.

Built-In Asset Transfer (in-band)

The primary function of any cryptocurrency is the ability to transfer assets in-band. This allows players (and especially miners) to transfer funds to other players, which from a game theory perspective, falls in the category of games with transferable utility. Depending on the degree to which such payments can be conditioned, e.g., if they are only valid on a specific branch or after a particular event, this also provides some enforcement capabilities. It can thus be argued that cryptocurrencies inherently allow for side-payments/bribes as part of their design. This immediately crosses out all system models in mechanism design that are not capable of dealing with side-payments among players.

Moreover, if a cryptocurrency offers the possibility to create smart contracts that are expressive enough, it might be possible to agree on side-deals or soft-fork rule changes by using smart contracts as an internal binding commitment/enforcement layer.

Asset Transfer with External Resources (out-of-band)

Besides the ability to utilize a targeted cryptocurrency directly for side-payments, the fact that a large ecosystem of multiple cryptocurrencies currently exists opens up the possibility for out-of-band side-payments issued in other (cryptographically interlinked) cryptocurrencies. Hereby, expressive smart contract capable cryptocurrencies already provide a suitable platform to launch AIM attacks against other cryptocurrencies as highlighted in [JSZ⁺19], or Chapter 4 of this thesis.

Therefore, it also appears conceivable that AIM may be used to undermine security assumptions in closed, so-called permissioned blockchains and DLT by targeting the implicit incentive structures that are often ignored in classical system models yet are likely present in real-world systems. This implies that DLT in closed systems that do not explicitly support cryptocurrencies may nevertheless be vulnerable to (cross-chain) bribing attacks. One future research direction is hence the analysis and application of AIM in such environments and the question of how constrained scripting capabilities in DLTs have to be able to avoid these issues.

Generally speaking, as soon as a system either temporarily or permanently agrees (i.e., decides) on a particular state, this result can be used as input to another interlinked system providing incentives to reach an attacker desired decision in the targeted system. Hereby, the system models of both systems share some interdependencies and can either worsen or constrain the problem at hand.

Asset Transfer with New External Resources (out-of-band)

Depending on the concrete scenario, the required financial resources (the budget) to successfully execute bribing and AIM attacks can be quite large. For example, if deep-forks or Goldfinger attacks are concerned. Some AIM attacks can work around this by performing less intrusive attacks that do not require costly forks, by programmatically sharing the potential profits with miners, or by crowdfunding the required resources.

Besides these techniques, there also exists the possibility to generate new virtual resources which can be used to finance AIM attacks. For example, the pitchfork attack described in [JSSW18], or in Chapter 2 of this thesis, creates a new cryptocurrency whose market value can be used to subsidize the Goldfinger attack on the targeted system, which inherently is part of the new protocol.

Non-Finality (also of Transactions)

Cryptocurrencies based on Nakamoto consensus, which provide probabilistic guarantees that a certain transaction has occurred at some place in the canonical chain, are not suited to derive final decisions regarding the transfer of external assets/resources. The reason for this lies in the fact that, as soon as a merchant has accepted a payment via this system and handed over the goods, the creator of the transactions has no more value associated with the respective account/key-pair. Therefore, the creator can equivocate and in retrospect cryptographically sign different payment transactions promising almost all (already spent) assets to other entities in the system. This attack is theoretically risk-free as the assets have been spent already anyway, but in the best case, some of these assets can be regained if the right entities can be bribed with the promised rewards. This makes any fork-able system a non-ideal candidate for a base layer of any cryptocurrency where the finality of an asset transfer is a necessary design feature from the perspective of all users that want to accept payments.

Even if the target system offers finality but at the same time assumes economically rational actors, then also “final” agreement could be revocable. If sufficiently large bribes are offered in a funding cryptocurrency to a threshold of nodes in the targeted system, it might be possible to incentivize a safety violation and thus lead to an agreement failure. For example, if equivocation is more profitable due to bribes (also accounting for

slashing and economic punishment in the target system), then equivocation (and thus an agreement failure) becomes economically rational.

7.4 Economically Rational Block Selection

In this section we describe a simplified *blackbox approach* to abstract all game elements and the concept of incentives by a function that determines the expected extractable value (EEV) of a player. For an economically rational miner a block b is preferred, compared to a different block b' , if the expected extractable value (EEV) for i is higher. This simple relation is depicted in equation 7.19. For a more detailed definition and discussion of EEV we refer to [JSSW22a], or Chapter 6 of this thesis.

$$\text{EEV}_i(\mathcal{R}, b') > \text{EEV}_i(\mathcal{R}, b) \quad (7.19)$$

When we focus on this individual perspective from a player i , we use an abbreviated notation for referring to $\text{EEV}_i(\mathcal{R}, \dots)$:

$$\text{EEV}(b') > \text{EEV}(b). \quad (7.20)$$

With this definition of EEV we follow a *blackbox approach* in which all costs and expected future profits are abstracted by the function EEV. This function computes the expected extractable value (i.e., the utility of proposing or reaching this block for the respective play), while considering the probabilistic nature of NC and, thus, the likelihood of current and potential future payoffs, as well as accounting for potential losses of funds and missed opportunities. For more information on how to calculate the EEV, we refer to [JSSW22a, JSSW22b], or the respective Chapters 5 and 6 of this thesis. If a player i is assumed to act economically rational, she will select the block which promises the highest expected extractable value. As we assume the economically rational behavior of proposers, their actions are primarily motivated by the expected gain in the respective resource(s). From the perspective of other actors, the resulting behavior can also be viewed as an attack on their individual gains. Therefore, we might also use the term *attack* if appropriate, especially later when we describe methods for third parties to incentivize the respective behavior, e.g., through bribes as part of algorithmic incentive manipulation. This should emphasize that the economically rational behavior of one party might lead to losses of another party and thus can be termed an attack from the perspective of the latter (cf. [JSSW22b] and Chapter 5).

But how does a proposer or miner construct or identify the most profitable block? Per definition, there can only be finitely many transactions in a block (determined by the block size \aleph), and a transaction is also required to be finite. Thus, depending on the actual size of a transaction, there can be a vast but necessarily finite number of potentially valid

transactions for any block. Still, there can only be \aleph valid transactions in the next block. Given the potentially huge search space and the open nature of the protocol, a natural question is how to select a sequence of transactions for the next block to determine the next operation. To answer this question, the set of transactions that are available to a certain player, the incentives of players as well as and their expected extractable value (EEV) from a certain sequence of transactions is relevant.

7.4.1 The Set of Available Transactions for Blocks

Block proposers have to select their next block based on their individual set of available well-formed transactions. We define the set of all well-formed transactions available to a player i as \mathcal{T}_i . The critical aspect here is that not every player can create every well-formed transaction. In practice, this is the case because it is infeasible to forge the cryptographic signatures required for the transactions.

Definition 28 (\mathcal{T}_i). The set of all well-formed transactions available to player i consists of:

- The set of all well-formed transactions collected by i , e.g., the mempool. These transactions have been created by other players that are not i , hence we denote them by $-i$, as a commonly used notation in the area of game theory. The transactions in this set cannot be created by i . The set of all well-formed transactions collected by i is denoted by \mathcal{T}_i^{-i} .
- The set of all well-formed transactions player i can create on his own, denoted by \mathcal{T}_i^i .

$$\mathcal{T}_i := \mathcal{T}_i^{-i} \cup \mathcal{T}_i^i \quad (7.21)$$

The set \mathcal{T}_i consists of all well-formed transactions available to player i , consisting of collected transactions that other players have created, denoted by \mathcal{T}_i^{-i} , in addition to all well-formed transactions player i can create by himself, denoted as \mathcal{T}_i^i .

Depending on the set of well-formed transactions available to a given player i , certain future blocks become theoretically reachable for this player or stay unreachable since the respective valid operation to reach it cannot be created. In turn, this means that the set of all potentially constructible valid operations changes with the set of available transactions. In other words, creating a blockchain is selecting a path through the set of all possible valid blocks \mathcal{O} of NC. Thereby, the set of the possible valid future blocks $\vec{\mathcal{O}}$ changes with each step, i.e., block. The set of possible valid future blocks $\vec{\mathcal{O}}$ depends on the set of available transactions \mathcal{T} , i.e., more available transactions in the mempool unlock a larger set of possible next blocks. Suppose now some actor published a new transaction tx , which exploits an arbitrage opportunity and directly bribes the miner of the next block for including it, the set of available well-formed transactions changes

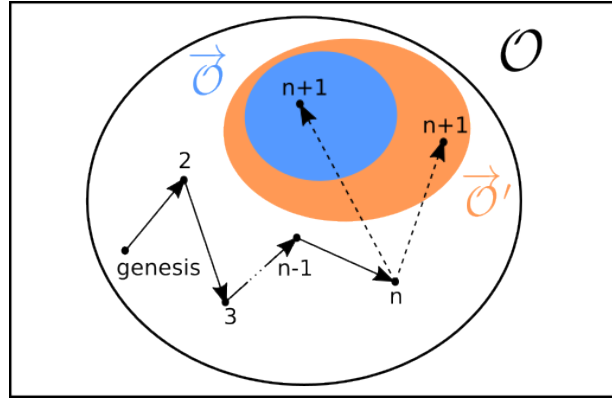


Figure 7.1: A blockchain is depicted as a specific sequence of n blocks within the set of all possible valid blocks. Starting from the latest block at height n , two potential next blocks in two different sets of valid future blocks are depicted. These two sets differ depending on the available set of transactions, i.e., $\vec{\mathcal{O}} = \text{FUTURE}(b_n, \mathcal{T})$ and $\vec{\mathcal{O}}' = \text{FUTURE}(b_n, \mathcal{T}')$, where $\mathcal{T} \subset \mathcal{T}'$ and thus $\vec{\mathcal{O}} \subset \vec{\mathcal{O}}'$.

to \mathcal{T}' . This, in turn, changes the set of reachable potentially valid future blocks to $\vec{\mathcal{O}}'$. Figure 7.1 shows such an example.

This example illustrates that an attack can influence the choice of the next block by an economically rational miner by publishing new transactions \mathcal{T}' such that a desired block gets more valuable for miners. Economically rational proposers will certainly not participate in such an attack if the existence of a more profitable block can be ruled out from his perspective:

$$\neg \exists b'_{n+1} \in \text{FUTURE}(b_n, \{\forall b \in \mathcal{O} | \text{TXSEQ}(b)\}), \text{EEV}(b'_{n+1}) > \text{EEV}(b_{n+1}). \quad (7.22)$$

If this does not hold, then there exists a set of transactions that leads to a more profitable block than b_{n+1} . Therefore, bribing attacks could potentially be feasible. For example, this means that mining a block with the desired ordering of some arbitrageur could be more profitable than mining a block without this specific transaction or set of transactions. If the resulting ordering is also more profitable for the issuer of the transaction, then this is a win-win situation. Although, the problem in practice is that if the respective guaranteed profit transaction becomes visible to other profit-seeking players, this transaction is vulnerable to front-running, leading to the observed behavior of bots [DGK⁺20, ZQC⁺21]. Note that, in such a bidding game, the achievable rewards of the miner increase while the achievable profits of the bidders decrease.

The same mechanism can also be used to incentivize miners to facilitate a double-spending attack. For example, by offering some of the achievable rewards as bribes. In such a double-spending attack, the block height x of the block to extend would be determined by the security parameter of the victim s.t. $x = n - k_V$. Moreover, the transactions in the respective blocks, which should become stale would again become part of the mempool and can be included in new blocks. Except the transaction tx , which should be double-spent and must therefore be excluded in the new sequence $\text{TXSEQ}((b_{n-k_V}, \dots, b_n) \setminus tx$.

Instead, a new conflicting transaction tx' is issued and part of the set of transactions \mathcal{T}' that is available for inclusion. If \mathcal{T}' also contains some bribing transaction(s) \mathcal{T}_B that are only available in the attacker-preferred branches/chains, an economically rational miner i could prefer mining on the double-spending branch. Conversely, if there does not exist any set of transactions such that blocks can be constructed that offer more expected extractable value compared to b_{n+1} , then an economically rational miner will not engage in any such attack as for him b_{n+1} is already the most profitable option.

$$\begin{aligned} &\forall b_{n+1} \in \text{FUTURE}(b_n, \mathcal{T}), \\ &\neg \exists b'_{x+1} \in \text{FUTURE}(b_x, \{\forall b \in \mathcal{O} | \text{TxSEQ}(b)\}), \text{EEV}(b'_{x+1}) > \text{EEV}(b_{n+1}). \end{aligned} \quad (7.23)$$

This again implies that the function EEV also accounts for potential losses due to the fork, e.g., of already contributed blocks, received payments, or drops in the exchange rate. This is analyzed in [JSSW22a, JSSW22b] as well as in Chapters 6 and 5. For the attacker, incentivizing deep-forks is profitable if the rewards from the double-spend are greater than the issued bribes. For the potentially bribable miners, the offered bribes have to be higher than their previously expected extractable value.

Those two examples differ regarding their level of interference with the rules of the system. In the first case, no fork is required, whereas in the case of a double-spend a fork of sufficient length is required. For a classification of different attacks regarding their level of interference with the rules of the system, see Appendix A.3.

7.4.2 Protocol Compositions using Adversarial Games

Apart from identifying profitable revenue opportunities on their own, proposers/miners can also be incentivized to propose certain attacker preferred blocks through AIM using bribes. Thereby, they do not have to search for the most profitable block themselves, but instead, they rely on a block provided by an attacker, which also carries an above-average reward. By following this approach, proposers voluntarily participate in the attack carried out by the initiator.

For example, an attacker can offer some of the achievable rewards from the attack as bribes to complacent proposers under the condition that a specific block or a series of blocks is proposed. As such attacks might also lead to losses of already contributed blocks, received payments, or drops in the exchange rate due to malicious activity, the function calculating the expected extractable value (EEV) for the proposer needs to account for that. In a multi-resource scenario, drops in the exchange rate can be avoided by having stakes in other non-affected or positively affected resources. For a discussion of this multiple-resources scenario see [JSSW22a, JSSW22b], or Chapter 6 and 5.

Losses of already contributed blocks can be avoided by compensations through bribes or larger rewards. In general, the ability to accurately reward compliant behavior automatically by using smart contracts (such as described in [JSZ⁺19, WHF19]), or

Chapter 4 and 5) reduces the costs of the attack drastically, compared to in-band reward methods in cryptocurrencies where this is not possible. In the UTXO model of Bitcoin, for example, the paper [KNW20] shows that immediate payoffs are preferred by miners if their hashrate is small, compared to larger rewards later on when payouts are only received later if all selected miners comply.

If out-of-band rewards are accepted by bribees, the situation changes drastically as sophisticated attacks which accurately reward compliant behavior can be constructed in external systems. This has been practically demonstrated in [JSZ⁺19] (Chapter 4 and Appendix A.6) and is generalized in Section 7.6.

We now want to analyse how and under which conditions adversarial games, targeting cryptocurrencies based on NC, can be constructed to facilitate algorithmic incentive manipulation. Thereby we use out-of-band payments to incentivize certain actions of economically rational miners in a way that is profitable for all compliant parties.

7.5 Adversarial Game to Manipulate Transaction Ordering (no-fork)

In the following, we describe adversarial games to facilitate algorithmic incentive manipulation. We begin with an adversarial game (AG) to incentivize a certain transaction ordering. This variant of the game represents an unintrusive optimization method for participating miners.

For this first scenario, we assume an attacker Eve (E), which wants to manipulate the ordering in a currently ongoing NC game by manipulating block proposals to her advantage. In contrast to other “classical” double-spending/private chain attacks, no revision (i.e., no fork) of the blockchain is required for such an attack. The attack can be started at any point in time and targets an unconfirmed transaction sequence τ , which includes the transactions tx_1 and tx_2 in this order in the suffix of sequence τ . We denote this as $tx_1 < tx_2$. In this case, transaction tx_1 comes from some other actor V , who is the victim in this case. The respective transactions have not yet been mined and thus have not been included in a block. Eve wants to replace this sequence with a sequence τ' , in which the transactions are reordered s.t. $tx_2 < tx_1$.

To do so, Eve starts a new adversarial game called ORDERGAME (Π') to influence the creation of the next block b_{n+1} , in the targeted system Π . As soon as the transactions in question are published and known by Eve, Π' is initialized. Without such a game, at this point in time, every miner would create his individual block proposal, which might look like $\bar{b}_{n+1}(\dots, tx_1, tx_2, \dots)$. This would create a chain of blocks c , where $\tau \prec \text{TxSEQ}(c)$, as soon as the topmost block has been mined. To create this block proposal \bar{b}_{n+1} a miner would include a coinbase/reward transaction tx_{reward} and append the new payment transactions to be mined after that. If lucky, the created block in sequence c then possibly becomes confirmed later, s.t. the miner can collect his block reward, including fees from transaction tx_{reward} in his previously created block $b_{n+1}(tx_{reward}, \dots, tx_1, tx_2, \dots)$.

If there now is a proposal chain \bar{c}' , where the head of the chain is $\bar{b}'_{n+1}(\dots, tx_2, tx_1, \dots)$, which provides higher expected profit than the original proposal chain \bar{c} , it can be assumed that economically rational miners would prefer this one over the other. To achieve this, the ORDERGAME pays out funds to the miner of a confirmed block b'_{n+1} if this block really includes the transactions $(\dots, tx_2, tx_1, \dots)$ in this particular order. As this attack can be executed multiple times in succession, it is sufficient to describe and analyze it for the ordering of the next block at height $n + 1$.

The remaining question for a successful attack is how Eve can specify transaction tx_{reward} without knowing who will mine the respective block. There are two possible solutions to this: The first approach is to add oneself (Eve) as a beneficiary in the tx'_{reward} transaction and compensate the respective bribed miner of the attacker desired block without a reward in the adversarial game Π' . In Π' Eve can pay out the compensation for the reward in Π together with the additional bribe. This approach is used in this section. The second approach is introduced in Section 7.6 where it is used in a different attack scenario.

An example attack, including concrete transaction sequences and the resulting chains, is listed below. Let's assume the current state of the blockchain and the associated ordered transaction sequence is as follows:

$$c_n = (b_1(tx_i), \dots, b_n(tx_j)) \quad (7.24)$$

$$\text{TXSEQ}(c_n) = (tx_i, \dots, tx_j) \quad (7.25)$$

A miner could now extend this sequence as shown in τ . The resulting sequence τ would then be used to create a new block template \bar{b}_{n+1} which is used as input to some mining functions to create the chain c_{n+1} .

$$\tau = (tx_i, \dots, tx_j, tx_{reward}, \dots, tx_1, tx_2, \dots) \quad (7.26)$$

$$c_{n+1} = (b_1(tx_i), \dots, b_n(tx_j), b_{n+1}(tx_{reward}, \dots, tx_1, tx_2, \dots)) \quad (7.27)$$

If an attacker now wants to ensure that the resulting transaction sequence contains tx_2 before tx_1 he either has to mine the next block, or incentivize other miners to include these transactions in this particular order. To do so he has to ensure the transaction sequence τ' is mined to create a chain resembling c'_{n+1} .

$$\tau' = (tx_i, \dots, tx_j, tx_{reward}, \dots, tx_2, tx_1, \dots) \quad (7.28)$$

$$c'_{n+1} = (b_1(tx_i), \dots, b_n(tx_j), b'_{n+1}(tx'_{reward}, \dots, tx_2, tx_1, \dots)) \quad (7.29)$$

Due to the probabilistic nature of NC, he has to define and wait for a confirmation period k_E after which he considers the desired sequence of transactions confirmed.

$$c'_{n+1+k_E} = (b_1(tx_i), \dots, b_n(tx_j), b'_{n+1}(tx'_{reward}, \dots, tx_2, tx_1, \dots), \dots, b_{n+1+k_E}(\dots)) \quad (7.30)$$

7.5.1 Assumptions of the AG OrderGame

In this section, we explicitly highlight all necessary assumptions for the attack. Some of these assumptions will later be lifted when their individual effect is analyzed in greater detail. Our basic assumption is that some fraction of miners is economically rational with respect to some set of resources \mathcal{R} . The fraction of economically rational miners, as well as the number of resources in \mathcal{R} , is dependent on the analyzed scenario and thus can vary depending on the respective case.

Assumption 1 (Economically Rational players). Some fraction of miners in the targeted NC game is economically rational with respect to some set of resources \mathcal{R} .

As in other models [GKL15], we simplify by requiring the difficulty in PoW cryptocurrencies to be constant. Thereby, the longest chain is also the heaviest chain in terms of difficulty.

Assumption 2 (Constant difficulty). For NC based on PoW, the difficulty for the whole duration of the game is constant.

To correctly payout rewards and compensations in the adversarial game Π' , it is required that the players in the targeted Π are able to assign information to receive payments to their attacker desired actions in Π . Otherwise, it would not be possible to correctly reward bribed players in Π , and as a result, no player in Π would have an incentive to accept bribes as it could not be ensured that they really receive them.

Assumption 3 (Attributability). Blocks are uniquely attributable to addresses/accounts of (pseudonymous) players to provide rewards.

To prevent corner cases where payouts in Π' are handled incorrectly, both games are assumed to be synchronized and that there is no network delay. This, for example, makes it impossible that a chain of blocks is submitted to Π' without being submitted to Π as well since players in Π' (including Eve) can always forward any received blocks to Π in time. For more details regarding the synchrony requirements between the two games see Section 7.5.4.

Assumption 4 (Synchronous communication). All messages (blocks and transactions) are received by all players before a known upper bound Δ . This does not mean that there necessarily exists a block in every interval. If there is a block in a certain interval, it is received by all players before the end of that interval.

Assumption 5 (No network delay). All messages are transmitted without any network delay.

With assumption 6, it should be ensured that all miners know that an adversarial game exists and are able to understand and verify its rules and payout scheme.

Assumption 6 (Perfect information regarding AG). All miners have perfect information regarding the adversarial game.

With assumption 7, it should be ensured that at least two payload transactions are available so that the ordering can become relevant.

Assumption 7 (Transaction availability $|\mathcal{T}_i \setminus \text{TxSEQ}(\text{CHAIN}(b_n))| \geq 3$). The number of transactions available to i , which have not yet been included in a block known to i , is at least $|\mathcal{T}_i \setminus \text{TxSEQ}(\text{CHAIN}(b_n))| \geq 3$, one of which is the payout transaction.

7.5.2 Rules of the AG OrderGame

In this attack, the attacker Eve (E) creates a new adversarial game Π' called ORDERGAME, which is defined by the 5-tuple $\Pi' = \langle \Pi, b_n, \tau', \epsilon, k_E \rangle$.

Hereby, the arguments to the game Π' initiated by E are: The specification of the original/targeted cryptocurrency, which itself is a NC game Π . This specification contains all necessary predicates, functions as well as the genesis block of Π . The current latest block b_n in the targeted game Π . The desired sequence of transactions τ' , including the transactions which have yet to be mined in the given order, as well as the offered bribe ϵ . Of course, the structure of the adversarial game has to plausibly assure that the offered reward ϵ is indeed paid out if Π' terminates. Depending on the concrete implementation as well as the computational capabilities of the targeted system, this payment can happen in-band (i.e., in the targeted cryptocurrency Π) or out-of-band (e.g., in a different cryptocurrency). The latter requires that there is more than one resource economically rational miners care about, i.e., $|\mathcal{R}| \geq 2$. In both cases, the ORDERGAME can be implemented as a smart contract on Ethereum, as demonstrated in [JSZ⁺19]. Either to attack Ethereum itself or another cryptocurrency with the help of Ethereum.

Moreover, a security parameter k_E is provided, which specifies the length of the suffix, after which the state of Π is considered acceptable in Π' . The adversarial game Π' has two termination conditions. The outcome of Π' is determined by the condition which happens first.

Termination condition I (success)

A player of Π' provides the game with a valid input sequence c' of length at least $n+1+k_E$ confirming the adversarial sequence of transactions τ' is part of the current state of Π .

Therefore, the following conditions hold:

$$b_n \prec \text{HEAD}(c') \wedge \quad (7.31)$$

$$\text{LENGTH}(c') \geq n + 1 + k_E \wedge \quad (7.32)$$

$$\text{VALID}(c') \wedge \quad (7.33)$$

$$\tau' = \text{TXSEQ}(b'_{n+1} \in c') \quad (7.34)$$

In other words, c' contains the desired list of transactions τ' , where tx_2 occurs before tx_1 .

If Π' is constructed using out-of-band payments, the chain c has to be submitted to AG before some player provides Π' with *any* other valid input sequence c of length $n + 1 + k_E$, where $\tau' \not\prec \text{TXSEQ}(c)$. This is required to not reward old/stale chains from Π in Π' . More information on the required synchrony between Π and Π' can be found in Section 7.5.4. If a valid attacker-preferred sequence of transactions of length $n + 1 + k_E$ has been submitted, the game rewards the miner of block b_{n+1} with an additional ϵ , i.e., with $r_{\text{reward}} + \epsilon$ in total, as also the rewards coming from payout transactions have to be compensated. This is necessary since the payout transaction in tx_{reward} in τ' is paid to Eve instead (in the original game Π). After the reward of Π' has been paid, the game terminates. The sequence c , which has been submitted to Π' , can then be replayed in Π as the new longest chain.

Termination condition II (fail)

A player of AG ORDERGAME provides the game with any other *valid* input sequence c of length at least $n + 1 + k_E$, where the following conditions hold:

$$b_n \prec \text{HEAD}(c) \wedge \quad (7.35)$$

$$\text{LENGTH}(c) \geq n + 1 + k_E \wedge \quad (7.36)$$

$$\text{VALID}(c) \wedge \quad (7.37)$$

$$\tau' \neq \text{TXSEQ}(b'_{n+1} \in c') \quad (7.38)$$

Then the AG terminates without paying out any rewards. The sequence c can also be submitted to AG by Eve if observed in the original game Π , and the AG has not yet terminated, i.e., if this is the first sequence of length, at least $n + 1 + k_E$.

7.5.3 Technical Feasibility and Computability of the AG OrderGame

As we are primarily interested in analyzing practical scenarios, we focus on instances of NC with a finite number of steps played and a finite set of players. We call these *finite chain instances*. To evaluate the practicality of adversarial game attacks on such instances, we analyze the computational feasibility of such scenarios, as well as the conditions for their profitability. We first show that adversarial game attacks on finite chain instances necessarily are technically feasible and efficiently verifiable in the out-of-band case. The feasibility of the in-band case depends on the capabilities of the targeted platform. We

then analyze the required assumptions s.t. ORDERGAME attacks on the NC game can be made profitable for economically rational miners.

Note that we are not interested in the complexity of the mining process itself, only in the evaluation and verification of its rules. Generally, for PoW, the process of mining can be split into two parts:

- *Block Mining*: This refers to spending a scarce resource, such as finding the solution to an intentionally hard puzzle, as done with PoW.
- *Block content composition*: This refers to the optimization problems to generate the maximum reward with a new block proposal.

Even for cryptocurrencies without smart contracts, block content composition is an NP-hard¹⁰ problem. To avoid defining the exact complexity and hardness of the whole mining process, including the associated optimization process, we focus on evaluating the rules given a set of blockchains, i.e., the evaluation of the predicate $\text{VALID}(\cdot)$ and the functions $\text{EXECUTE}(\cdot)$ and $\text{MAIN}(\cdot)$. This ensures that blockchains can be verified and the longest one can be selected. Note that, in a model where the difficulty is constant, the longest chain is automatically also the heaviest chain and thus the main chain.

In other words, we focus on the computability of evaluating the rules of NC as well as the computability of the rules of the adversarial game to attack it. As in practice, we can only observe *finite chain instances*, we will only consider finitely many finite blockchains and finite executions of the adversarial game that is aimed to attack it.

In an ORDERGAME attack, Eve is required to construct a desired/malicious sequence τ' of transactions. Therefore, Eve would have to create a block proposal \bar{b}_{n+1} with her desired sequence of payment transactions. In our example, this would result in an ordering where tx_2 comes before tx_1 . As \bar{b}_{n+1} is a block proposal, the rational miner who will successfully mine this proposal cannot readily be determined. Therefore, the first transaction in the block which determines the reward in Π (i.e., tx_{reward}) cannot be defined correctly upfront by Eve. In this section, we work around this by letting Eve define a transaction sequence where the payout of tx'_{reward} goes to her, while the adversarial game compensates the miner for the missing payout reward.

To analyze the computability of the described ORDERGAME attack and its adversarial game targeting finite chain instances, we first start by providing the necessary definitions.

Definition 29 (Finite chain instance). An instance of infinite NC with finitely many players, which has been executed for finitely many steps, i.e., blocks, is called a *finite chain instance*.

¹⁰cf. <https://freedom-to-tinker.com/2014/10/27/bitcoin-mining-is-np-hard/>

The focus on finite chain instances implies that there can only be finitely many competing blockchains. As a result, only finitely many chains have to be checked to determine the longest and thus the canonical chain. This is the case because we are in a model with constant difficulty, in which the longest chain equals the heaviest chain.

In our ORDERGAME attack approach, Eve defines a block template, e.g., $\bar{b}_{n+1}(tx'_{reward}, t_2, t_1)$, in the adversarial game Π' , containing a transaction tx'_{reward} paying the block reward in Π to Eve. In this case, Π' has to reward the adversary and compensate the reward for this block (including fee) to the respective miner directly in the adversarial game. As we are interested in the practical feasibility, we focus on attacking a finite number of steps of a specific instance of NC. Therefore, we define an efficiently verifiable game as follows.

Definition 30 (Efficiently verifiable game). A game is efficiently verifiable if its rules can be evaluated, for a finite number of steps and players in polynomial time.

Lemma 1 (The NC game is efficiently verifiable). Finite chain instances of NC are efficiently verifiable, i.e., its predicate and functions, except the function PROPOSE(\cdot), can be evaluated for a finite number of chains, steps, and players in polynomial time.

The NC game is efficiently verifiable. Per Definition 30, we are only interested in instances with finitely many steps. Therefore, the longest chain and thus all other chains can only have finitely many blocks. This also implies that there can only be finitely many competing chains that might have been produced. As a result, an *InTree* of all blockchains with a common genesis block can only have finitely many vertices, i.e., blocks. To prove that NC is efficiently verifiable, we now show that the time complexity, measured in execution steps, of all functions and predicates in NC is polynomially bounded:

- EXECUTE(\cdot) runs on a given state as well as a block. Per definition, the function is polynomially bounded and required to terminate to either return a response or \perp . Thus, per definition, EXECUTE(\cdot) clearly runs in polynomial time for some input m and some constant c_1 , i.e., $O(m^{c_1})$.
- VALID(\cdot) invokes EXECUTE(\cdot) and runs on a chain of blocks of length n . Furthermore, it might require another constant c_2 . Therefore, it has time complexity $n^{c_2} \cdot \text{EXECUTE}(\cdot)$. Thus VALID(\cdot) runs in polynomial time as well, i.e., $O(\max(n, m)^{c_1+c_2})$.
- MAIN(\cdot) runs on a given *InTree*, thus the complexity of finding the longest path is linear in the number of vertices and edges [CLRS01]. Let N be the number of vertices. As every block only has exactly one predecessor and the genesis block has none, there are exactly $N - 1$ edges in the *InTree*. Thus the complexity of finding the longest path is $O(N + (N - 1))$ which simplifies to $O(N)$. Under the

assumption that the $InTree$ is constructed at each client and only valid blocks are added to the $InTree$, the longest path represents the canonical chain. If also invalid blocks (except for the correctness of the cryptographic hash) can be included in the $InTree$, the function still runs in polynomial time. As shown, checking if the initially found longest path, i.e., chain, is valid, runs in polynomial time. If the discovered longest path is not valid, the last node (block) of the path is removed from the $InTree$, and the algorithm is executed again. In the worst case, this results in N executions, checking in the worst case a chain of length N in the first round. Thus the worst case complexity is $O(N) \cdot O(N) \cdot O(\max(N, m)^{c_1+c_2})$, which simplifies to $O(\max(N, m)^{2+c_1+c_2})$.

As all necessary functions and the predicate $VALID(\cdot)$ of NC run in polynomial time, we have shown that NC is an efficiently verifiable game according to Definition 30. \square

Now we have to show that if the NC game can be considered *efficiently verifiable*, then an adversarial ORDERGAME can be constructed to attack the incentives of NC that are also efficiently verifiable and eventually terminates if the targeted system makes progress. To keep the proof generic for adoption in Section 7.6, we only refer to the ORDERGAME as adversarial game (or abbreviate it by AG) for the rest of this section.

Theorem 3 (There exists an AG targeting a NC game that is efficiently verifiable and eventually terminates). An adversarial game (AG) targeting a finite chain instance of the NC game is efficiently verifiable and eventually terminates if the targeted instance of a NC game is efficiently verifiable and eventually makes progress.

To prove that it is indeed technically and computationally feasible to construct an AG for Π , we first prove that such an AG is efficiently verifiable if the underlying NC game is efficiently verifiable. Then we show that the AG eventually terminates if the underlying NC game eventually makes progress.

Lemma 2 (The AG is efficiently verifiable). If a NC game is efficiently verifiable, i.e., its rules can be computed in polynomial time, then an adversarial game (AG) aimed to attack a finite chain instance of the NC game is also efficiently verifiable.

The AG is efficiently verifiable. The blockchain up to block b_n , as well as any other blockchain of length n , is necessarily finite (as long as n is finite which is the case per definition). Since NC is efficiently verifiable and the function $VALID(\cdot)$ has polynomial runtime, there must exist an efficient algorithm deciding if a chain of blocks c' is valid in $1 + k_E$ additional steps of the original NC protocol. The helper functions $LENGTH(\cdot)$, $PREV(\cdot)$, and $HEAD(\cdot)$, can also run in at least n steps (for traversing the given chain) and may require some constant exponent. Therefore, they also clearly run in polynomial time. Selecting $b'_{n+1} \in c'$ can be done by executing $PREV(\cdot)$ for k_E times and therefore also runs in polynomial time. Additionally, we have to show that determining $b_n \prec HEAD(c')$, and $\tau' = TXSEQ(b'_{n+1})$, is also doable in polynomial time. Let x be the length of the

smaller sequence of transactions or blocks respectively. Then identifying if a sequence is a prefix or equivalent of another sequences is clearly doable in polynomial time as iterating and comparing two lists of transactions or blocks is bound by the (necessarily finite) number of elements of the smaller sequence and thus a polynomial $P(x)$. The same holds for determining $b_n \prec c'$, or $b_n \not\prec c'$. Therefore, the AG is efficiently verifiable if NC is efficiently verifiable. \square

Lemma 3 (The AG eventually terminates). If a targeted NC game eventually makes progress, an attacking adversarial game (AG) will eventually terminate.

The AG eventually terminates. If NC eventually makes progress, some chains will eventually reach $n + 1 + k_E$ blocks. Since we are only interested in attacking a specific instantiation of NC, and thus a finite sequence of steps of NC, this implies that there can only be finitely many chains of blocks for NC (i.e., forks) at any given point in time, which in turn can be submitted to the AG. As the AG terminates when the first valid chain of length $n + 1 + k_E$ is found, only finitely many blockchains have to be validated by AG, as AG terminates on the first valid chain of this length. Therefore, if NC makes progress, some chains will eventually reach the required number of blocks and can be submitted to the AG, which will eventually terminate. \square

There exists an AG that is efficiently verifiable and eventually terminates. From our Lemma 2, it follows that the AG is efficiently verifiable if the targeted NC game is efficiently verifiable. From Lemma 1, we know that the NC game is indeed efficiently verifiable and so must be the AG to attack it. Lemma 3 shows that the AG eventually terminates if the targeted NC game eventually makes progress. Therefore it follows that an AG is indeed efficiently verifiable and eventually terminates if the targeted finite chain instance of the NC game is efficiently verifiable and eventually makes progress. \square

7.5.4 Practical Considerations for the OrderGame

The possibility of constructing an AG (such as the ORDERGAME) with in-band rewards depends on the capabilities offered by the targeted system. For Ethereum, for example, it is possible, as shown in [JSZ⁺19], while for Bitcoin, it is not possible as the ordering of transactions is not accessible from within the system.

For the out-of-band case, the proof of Theorem 3 shows that it is technically feasible to construct an efficiently verifiable AG which is running in parallel to the original NC game. Provided that the out-of-band rewards for the AG can be credibly paid out and are available in a resource $R \in \mathcal{R}$ economically rational players of the targeted NC game care about. If this is the case, then this composition could interfere with the incentives of such players in the NC game. If the state and all messages of the targeted NC game are publicly observable, as it is the case with so-called *permissionless* cryptocurrencies, it is not feasible to prevent the creation of such external adversarial games as all the data to construct them is publicly available.

An interesting question and subject for future work are the exact communication and synchrony requirements between those two games, as well as the impact and inheritance characteristics of the chosen consensus approach (finality vs non-finality, permissionless vs permissioned). In the following we will highlight key challenges and possible solution approaches from a practical perspective.

Synchrony between the two games and Finality

From an attacker's perspective, targeting a cryptocurrency that does not offer finality also has some disadvantages. If the targeted system does not provide finality, also the attacking system cannot offer finality without risking false payouts. In the simplest case, the targeted game Π would actually agree on states and thereby offer some form of finality, i.e., a resulting agreed-upon state. This final state can then be used as input to the adversarial game Π' . Therefore, Π' could also offer finality and decide¹¹ based on the decision made in Π . The remaining challenge would be to inform sufficiently many players of Π about the potential rewards in Π' before they make their final decision about the respective state in Π , which should be influenced by Π' .

Unfortunately, NC never actually agrees and thus does not offer finality (as described in Section 7.3). Therefore, it is necessary to prevent the AG Π' from making a wrong decision regarding the actual state of the targeted NC game Π . As in the original NC protocol, the probability that some sequence remains part of the canonical chain can be adjusted by the choice of the security parameter k_E . Hereby, it is even possible that the original global security parameter of the NC game is smaller than the parameter used in the AG, i.e., $k < k_E$. So in terms of the decision, the AG can only provide probabilistic guarantees if the targeted system only provides probabilistic guarantees. But how to ensure that the same inputs are used in both games?

To prevent that some chain of length $N = n + 1 + k_E$ is only submitted to the AG Π' , but not to the original NC game Π , some form of relay between those two systems is required. As Eve wants her attack to have an effect on Π , she is assumed to replay any chain c' sent to Π' to Π . If there is no network delay (Assumption 5), every input to Π' can immediately be replayed and used as input to the original NC protocol Π as well. Moreover, Eve has a strong incentive to prevent the AG Π' from paying out unnecessary rewards. Therefore she will relay current information on the state of Π (i.e., non-attack chains c) to the AG as well. Note that it would not be relevant if Π' terminates after the same input has already been verified in Π , as long as all inputs received are processed in sequential order by Π' . So it is fine if Π' takes polynomially more time to compute the rules and terminates after Π has already produced a result for the same chain of blocks.

¹¹If the attacking system works by agreeing on the output (consensus decision) of the targeted system, then this agreement can of course only happen after the target system has agreed. Thus if the target system is asynchronous, also the attack has to be asynchronous (not necessarily the underlying system which for example hosts the attack smart contract). If the target system is synchronous, the synchrony bound regarding the output of the target system, from the perspective of the attacking system, needs to be before the attacking system can agree.

If we would not assume synchrony, and there would be some network delay, the Π' could be tricked into paying out bribes even though the required state has not occurred in Π . In the worst case, both systems are not synchronized, and Π' receives a chain of length N and terminates, while this specific chain will not become part of the canonical chain in Π . The reason for this lies in the nature of NC as a probabilistic system. Still, due to the properties of NC valid inputs (i.e., blocks), even from older rounds, will not be discarded but included in the *InTree* at their respective positions. Therefore, valid blocks always have some effect, such as advancing a stale branch to make further attacks more likely.

Let us now sketch some other approaches of how to deal with this aspect of synchrony between AG and the NC game in this context:

Always propagate changes in Π to Π' The first approach would be to tie the state and thus the outcome of Π' to the state of Π . This would for example be possible in case of an in-band attack if the descriptiveness of the target cryptocurrency allows the creation of Π' on the application layer of the target cryptocurrency. For example, if the target cryptocurrency supports expressive smart contracts such as in Ethereum, Π' can be constructed as a smart contract in the respective currency under attack, thereby all state changes in Π would immediately propagate to Π' , where it leads to the correct decision. Such an in-band attack based on smart contracts in Ethereum was described in [JSZ⁺19]. In this case, it is also not necessary to compensate the block rewards of miners participating in Π' and mining the block with the respective ordering.

Provide incentives to comply In this approach, Π' is constructed in a way such that it is more profitable to comply than to sabotage Π' . In our case, for example, mining and keeping $1 + k_E$ blocks secret to only submit them to Π' is less profitable than participating in the mining process of Π directly and only later if sufficient blocks have been found, submit them to Π' if this promises additional rewards.

Transfer the risk by cashing out Another approach would be to transfer the risk that the state of the target cryptocurrency changes in retrospect by introducing some additional delay to be able to cash out in Π as long as the attack chain is in the lead. For this to work, Π' has to be adapted in a way that a delta (Δk_{E_1}) between the highest block of the attack chain and the second longest chain without the attack must reach a certain value. If this delta (in terms of blocks) is reached, another phase starts in which there is a new delta Δk_{E_2} and some delay, after which the AG really terminates, and all rewards and bribes are paid if still no other longest chain is known. With this construction, there is now a time window between the advantage of Δk_{E_1} and the new delta Δk_{E_2} , in which the longest chain known is the attack chain. Therefore, Eve can use the respective chain and submit it to Π , where it will take the lead. If Eve now finds a merchant who has a fixed security parameter $k_{merchant}$ that fits between the current head of the chain and the new delta Δk_{E_2} , she can use this merchant to cash out her in-band profits from the attack and thus will keep them even if Π later reverts. Still, Π' would have paid out rewards unnecessarily, but Eve already has secured her gains and

thus can ignore this fact. This is comparable to the *let's go shopping approach* described in [JSSW22a], as well as in Chapter 6.

7.5.5 Profitability of the AG OrderGame

$$(p_A + p_V + p_B = 0)$$

To assess the general profitability, we first have to analyze what attributes are required, s.t. participating in Π' becomes economically attractive for economically rational miners in Π . In this setting, all miners are assumed to be economically rational, and the victim is assumed to be a user which does not launch a counter-attack. To analyze if economically rational miners would participate in the attack, the expected extractable value of a miner is higher when he participates in Π' than when he decides to continue to exclusively play the original NC game Π . Playing Π' offers a higher expected extractable value for miners if the cost of participating ($r_{participate}$) is smaller than the additional income from the bribe (ϵ). In this case, the costs might arise when submitting the final result c' to the smart contract implementing the logic of Π' . Since the costs for mining ($r_{miningcost}$) are the same with, as well as without the attack, they can be ignored. Therefore, if there exists an adversarial game Π' which offers an additional reward $\epsilon - r_{participate} > 0$ for a block proposal for a sequence τ' compared to a block proposal for a sequence τ , then an economically rational miner would prefer the sequence τ' and thus work on c' to be later able to submit it to Π' , as $p_i \cdot (r_{reward} + \epsilon - r_{participate}) > p_i \cdot r_{reward}$.

Profitability of the AG OrderGame for Eve

We now analyze the required conditions s.t. the attack becomes economically attractive for the economically rational attacker Eve. We then analyze under which assumptions launching an AG attack can still be profitable for the attacker. For the attack to be profitable for an attacker E , the expected extractable value for Eve from the ordering τ' , compared to τ , has to be higher than the paid bribe plus operational costs for launching the adversarial game, i.e., $\epsilon + r_{operational} < \text{EEV}(b_{n+1})$, where $\tau' \preceq \text{TXSEQ}(\text{CHAIN}(b_{n+1}))$. This can be the case, for example, when winning an auction or due to making a trade to exploit a market inefficiency as described in [DGK⁺20]. Then creating an adversarial game Π' to incentivize the ORDERGAME, offers a higher expected profit for the attacker in terms of cryptocurrency units of the targeted system than playing the original NC game.

Analysis for cases where not all miners are economically rational can be found in Chapter 4 as well as Appendix A.5 and A.6.

7.5.6 Countermeasures

What options does the victim V have to mitigate the attack of Eve? If V also has perfect information regarding the attack, he can initiate a counter-attacker that works in the same manner as the original attack, i.e., he immediately launches an adversarial game $\Pi'' = (\Pi, b_n, \tau, \epsilon_V)$, where $\epsilon_V > \epsilon$. Then there would be two opposing attacks running in

parallel, and the miners would have to pick the most profitable one. In one game, the attacker is Eve, and in the other game, the attacker is Vincent. This could lead to an arms race, in which the original attacker, Eve, also increases the rewards of her original attack s.t., again $\epsilon > \epsilon_V$, and so forth. In the end, this leads to a bidding game for the desired transaction ordering.

For a discussion of possible countermeasures, also see [JSZ⁺19, JSZ⁺21b], as well as the Chapters 4 and 3 of this thesis.

7.6 Adversarial Game to Exclude Transactions (near-fork)

For this scenario, we assume an attacker Eve (E), which does not want that a specific transaction tx_V from some victim V is included in the blockchain for a certain period of blocks $\ell > 0$, which denotes the duration of the attack. To do so, Eve starts the attack as soon as an unconfirmed transaction tx_V is broadcasted and would result in a transaction sequence τ , which includes the unwanted transaction. Eve now wants to replace this sequence with a sequence τ' , which excludes tx_V , such that τ' gets confirmed. Therefore, it has to be ensured that the transaction is not included within the next ℓ blocks. If $\ell = 1$, then the attack only lasts for one block. If $\ell > 1$ then the attacks need to sustain ℓ blocks to ensure the transaction tx_V will not be included in later blocks until block $n + \ell$, where n is the current block height. Therefore, Eve needs to provide a reward ϵ for each block from b_{n+1} to $b_{n+\ell}$ if the unwanted transaction tx_V is not included in those blocks. To incentivize this behavior, the adversarial game EXCLUSIONGAME is created.

In the following, we want to provide a list of example sequences and blockchains for the described attack. First of all, let's assume the current state of the blockchain is b_n , containing the sequence of transactions τ_m , i.e., $\text{TXSEQ}(b_n) = \tau_m$.

$$\tau_m = (tx_0, \dots, tx_m) \quad (7.39)$$

$$c_n = (b_1(tx_0), \dots, b_n(tx_m)) \quad (7.40)$$

When the targeted transaction tx_V is published, the unmodified blockchain would be c which includes the transaction in question.

$$\tau = (tx_0, \dots, tx_m, tx_{m+1}, \dots, tx_V) \quad (7.41)$$

$$c_{n+1} = (b_1(tx_0), \dots, b_n(tx_m), b_{n+1}(tx_{m+1}, \dots, tx_V)) \quad (7.42)$$

Eve now wants to exclude tx_V for the duration of ℓ blocks, which should result in a blockchain c' of length $n + \ell$ without tx_V in it. In the end, this chain will be confirmed

by some suffix sufficiently deep such that the security parameter (k_E) of Eve is satisfied.

$$\tau' = (tx_0, \dots, tx_m, \dots, tx_j) \quad (7.43)$$

$$c'_{n+\ell} = (b_1(tx_0), \dots, b_n(tx_m), \dots, b_{n+\ell}(\dots, tx_j)) \quad (7.44)$$

$$c'_{n+\ell+k_E} = (b_1(tx_0), \dots, b_n(tx_m), \dots, b_{n+\ell}(\dots, tx_j), \dots, b_{n+\ell+k_E}(\dots)) \quad (7.45)$$

We argue that this censorship attack is different compared to attacking the liveness property of a transaction. In comparison, the liveness property of a transaction ensures the sender that the transaction eventually gets included in the canonical chain. In contrast, we consider a transaction effectively censored when it is not included in a certain sender-defined state or until before a certain sender-defined position in the sequence of transactions. Therefore, transaction liveness is a weaker security property compared to transaction censorship, as the value of a transaction for the sender might depend on sender-defined inclusion requirements. For example, a profitable arbitrage or front-running transaction is worthless if another transaction is executed before. Another example would be the closing of a payment channel.

7.6.1 Assumptions of the AG ExclusionGame

All assumptions from Section 7.5.1 still apply. Moreover, for our initial description, we also introduce the following assumptions, which will be lifted and discussed in the context of counter-attacks. For now, these assumptions should simplify the profitability analysis of the attack, as no additional payments to miners have to be considered (besides regular block rewards).

Assumption 8 (Independent transactions). For the duration of the attack, all transactions are independent and thus do not build upon each other.

Assumption 9 (No payment transactions to miners). For the duration of the attack, i.e., from the block at position n to the end of the attack, miners do not receive any payment transactions, only reward transactions with the respective block.

7.6.2 Rules of the AG ExclusionGame

For the attack, Eve creates a new adversarial game Π' called `EXCLUSIONGAME`, which is defined by the 7-tuple $\Pi' = \langle \Pi, b_n, \text{EEXECUTE}(\cdot), \text{EVALID}(\cdot), \epsilon, k_E, \ell \rangle$.

Hereby, the arguments to the game Π' initiated by E are: The specification of the original/targeted cryptocurrency, which itself is a NC game Π . This specification contains all necessary predicates, functions as well as the genesis block of Π . The current latest block b_n in the targeted game Π . A new function `EEXECUTE`(\cdot) that only produces a response that is not \perp if a desired state is computed in the required block interval. This is only the case if the effects of the unwanted transaction have not occurred, i.e., if tx_V is not part of that respective block interval. For all other inputs not containing tx_V or not in the respective interval, the response messages from `EXECUTE`(\cdot) and `EEXECUTE`(\cdot)

are equivalent. The computed states of $\text{EXECUTE}(\cdot)$ and $\text{EEXECUTE}(\cdot)$ are the same for all inputs. Moreover, such as $\text{EXECUTE}(\cdot)$, the function $\text{EEXECUTE}(\cdot)$ is assumed to run in polynomial time. As a wrapper around the new execute function, a function $\text{EVALID}(\cdot)$ is used, which works the same as the original function $\text{VALID}(\cdot)$ but invokes $\text{EEXECUTE}(\cdot)$ instead of $\text{EXECUTE}(\cdot)$.

The offered bribe per block is denoted by ϵ , and the duration of the attack is given by ℓ . The parameter k_E denotes the security parameter of the attacker Eve and represents the waiting period after which the submitted attack sequence is considered confirmed by Π' .

Of course, the structure of the adversarial game has to plausibly assure that the offered reward ϵ is indeed paid out if Π' terminates. Depending on the concrete implementation, this payment can happen in-band (i.e., in the targeted cryptocurrency Π) or out-of-band (i.e., in a different cryptocurrency). In both cases, this can be ensured with the help of a smart contract, for example, demonstrated for Ethereum in [JSZ⁺19].

The adversarial game Π' has two termination conditions. The outcome of Π' is determined by the condition which happens first.

Termination condition I (success)

A player of Π' has to provide the adversarial game with a valid input sequence c' of length at least $n + \ell + k_E$, where the following conditions hold:

$$b_n \prec c' \wedge \quad (7.46)$$

$$\text{LENGTH}(c') \geq n + \ell + k_E \wedge \quad (7.47)$$

$$\text{VALID}(c') \wedge \quad (7.48)$$

$$\text{LENGTH}(c'_{n+\ell}) = n + \ell \wedge \quad (7.49)$$

$$\text{EVALID}(c'_{n+\ell}) \quad (7.50)$$

In other words, this submitted blockchain c' confirms a sub-chain $c'_{n+\ell}$ from block n to block ℓ , which does not contain the effects of the unwanted transaction tx_V , i.e., $\text{EVALID}(c'_{n+\ell}) = \text{TRUE}$. If this is the case, then the game rewards every miner from block $n + 1$ to block $n + \ell$, with an additional ϵ per block in the new game Π' . Since the attacker does not have to predefine the entire transaction sequence, i.e., the content of the blocks, the game does not have to compensate the respective miners for missing rewards in Π . Therefore, the total amount paid out for blocks in the range $[n + 1, n + \ell]$ is:

$$\sum_{i=n+1}^{n+\ell} \epsilon \quad (7.51)$$

After the rewards have been paid, the game terminates.

Termination condition II (fail)

A player of Π' provides the adversarial game with any other valid input sequence c of length at least $n + \ell + k_E$, where the following holds:

$$b_n \prec c \wedge \quad (7.52)$$

$$\text{LENGTH}(c) \geq n + \ell + k_E \wedge \quad (7.53)$$

$$\text{VALID}(c) \wedge \quad (7.54)$$

$$\text{LENGTH}(c_{n+\ell}) = n + \ell \wedge \quad (7.55)$$

$$\text{EVALID}(c_{n+\ell}) \quad (7.56)$$

In other words, a different blockchain c which does not satisfy $\text{EVALID}(c_{n+\ell})$ has been submitted before a desired chain c' has been submitted. Note that such a chain c can also be submitted to Π' by Eve if she observed this chain in Π and the AG has not yet terminated, i.e., if this is the first blockchain of length, at least $n + \ell + k_E$. For this AG, several different rewarding schemes can be imagined. In the following, we will describe two, namely *no compensation* and *effort-related compensation*.

No Compensation: If the attack was not successful, i.e., tx_V was included in the respective interval, the adversarial game terminates without paying any reward. This is comparable to the attack described in [KNW20] or the *pay-per-miner* variant in [WHF19].

Effort-related Compensation: In this case, the adversarial game accepts and extends one candidate chain $c'_{candidate}$, which is initialized with b_n . A block b_{new} is appended to $c'_{candidate}$ if the following conditions hold:

$$\text{PREV}(b_{new}) = \text{HEAD}(c'_{candidate}) \wedge \quad (7.57)$$

$$b_n \preceq (c'_{candidate}, c_{new}) \wedge \quad (7.58)$$

$$n < \text{LENGTH}\left((c'_{candidate}, c_{new})\right) \leq n + \ell \wedge \quad (7.59)$$

$$\text{VALID}\left((c'_{candidate}, b_{new})\right) \wedge \quad (7.60)$$

$$\text{EVALID}\left((c'_{candidate}, b_{new})\right) \quad (7.61)$$

After any chain c (of length, at least $n + \ell + k_E$) has been submitted to Π' , it terminates as described above. But now, if Π' has failed, it still compensates all blocks in $c'_{candidate}$ with r_{reward} , i.e.,

$$\sum_{i=n+1}^{\text{LENGTH}(c'_{candidate})} r_{reward}. \quad (7.62)$$

This ensures that blocks of complacent miners are compensated, even if the attack is unsuccessful, i.e., $c'_{candidate}$ did not become part of the canonical chain.

This compensation scheme could also be built in a way that monitors any block race, i.e., all submitted chains in the respective interval, and in the end, rewards the longest candidate chain.

Note that extending an already running attack is always possible by chaining different attacks. The important aspect is that already promised rewards must not be negatively affected, i.e., reduced. Increasing the overall budget for bribes is always possible for Eve. If Eve starts with a small ℓ_1 and increases the duration of the attack by adding another attack ℓ_2 and so forth till the overall ℓ is reached.

7.6.3 Technical Feasibility and Computability of the AG ExclusionGame

For analyzing the proposed exclusion attack, we are again interested in its computational feasibility. Analogous to Section 7.5.3, we first show that the AG EXCLUSIONGAME attack on a *finite number of steps* of the NC game is computationally feasible to construct under the defined conditions and assumptions.

To apply our original proof of Theorem 3, it is important that the newly introduced predicate EVALID(\cdot) and the underlying function EEXECUTE(\cdot) only introduce polynomially bounded runtime overhead. Moreover, for the *effort-related compensation* reward mechanism, we have to show that the AG still terminates. Therefore, the original proofs for the lemmas 2 and 3 to prove the Theorem 3 have to be adapted. To show that the EXCLUSIONGAME is efficiently verifiable and eventually terminates, we extend the original proofs as follows:

The AG is efficiently verifiable. The blockchain up to block b_n , as well as any other blockchain of length n , is necessarily finite, as long as n is finite. Since NC is efficiently verifiable and the function VALID(\cdot) has polynomial runtime, there must exist an efficient algorithm deciding if a chain of blocks c' is valid in $1 + \ell + k_E$ additional steps of the original NC protocol. The helper functions LENGTH(\cdot), PREV(\cdot), and HEAD(\cdot), can also run in at least n steps (for traversing the given chain), and may require some constant exponent. Therefore, they also clearly run in polynomial time. Additionally, we have to show that determining $b_n \prec \text{HEAD}(c')$, and $\tau' \not\prec \text{TXSEQ}(c')$, is also doable in polynomial time. Let x be the length of the smaller sequence of transactions. Then identifying if a sequence is a prefix of another sequences is clearly doable in polynomial time as iterating and comparing two lists of transactions or blocks is bound by the (necessarily finite) number of elements of the smaller sequence and thus a polynomial $P(x)$. The same holds for determining $b_n \prec c'$, or $b_n \not\prec c'$. Per definition, EEXECUTE(\cdot) runs in polynomial time. As EVALID(\cdot) executes EEXECUTE(\cdot) for $n + \ell + k_E$ times, it also runs in polynomial time. Therefore, AG is efficiently verifiable if NC is efficiently verifiable. \square

The AG eventually terminates. If the targeted finite chain instance of the NC game eventually makes progress, some chains will eventually reach $n + \ell + k_E$ blocks. Since we are only interested in attacking a specific instantiation of NC, and thus a finite sequence of steps of NC, this implies that there can only be finitely many chains of blocks for NC (i.e., forks) at any given point in time, which in turn can be submitted to the AG. As the AG terminates when the first valid chain of length $n + \ell + k_E$ is found, only finitely many blockchains have to be validated and tracked by AG, as AG terminates on the first valid chain of this length. Therefore, if NC makes progress, some chains will eventually reach the required number of blocks and can be submitted to the AG, which then will eventually terminate. \square

With the changed proofs for both lemmas, the original theorem 3 can again be proven as done in Section 7.5.3. As the REVISIONGAME attack also relies on the same techniques used and described in the EXCLUSIONGAME variants, i.e., a predicate EVALID(\cdot), no change to the proofs are required, besides adapting the length of the chain.

7.6.4 Practical Considerations for the ExclusionGame

If a variant of the EXCLUSIONGAME can be implemented using in-band payments depends on the capabilities of the target cryptocurrency. A variant of the EXCLUSIONGAME with *no compensations* and in-band rewards is possible to construct using smart contracts in Ethereum as demonstrated in [JSZ⁺19].

As with the ORDERGAME, the proof of Theorem 3 shows that it is possible to construct an out-of-band variant of the EXCLUSIONGAME that runs in parallel to the targeted NC game. Again this requires that the out-of-band rewards for the AG can be credibly paid out and are available in a resource $R \in \mathcal{R}$ economically rational players of the targeted NC game care about. If this is the case, then this composition could interfere with the incentives of such players in the NC game. If the state and all messages of the targeted NC game are publicly observable, as it is the case with so-called *permissionless* cryptocurrencies, it is not feasible to prevent the creation of such external adversarial games as all the data to construct them is publicly available.

The requirements and options regarding communication/synchrony requirements between those two games are the same as in the ORDERGAME.

7.6.5 Profitability of the ExclusionGame

In the beginning, we assume that there are only economically rational miners and the victim(s) do not have hashrate, i.e., $p_A + p_V + p_B = 0$. In this case, there are only economically rational miners, and the victim(s) (\mathcal{V}) have no hashrate. Therefore, the profitability of participating in the adversarial EXCLUSIONGAME (Π'), while mining blocks for the targeted NC game Π , can be ensured for the remaining NC miners if playing Π' offers a higher expected extractable value for NC miners than playing only the original NC game, i.e., if $\epsilon - (r_{participate} + \text{EEV}(tx_V)) > 0$. Therefore, a rational miner

would prefer the sequence excluding tx_V to be able to submit the resulting blocks to Π' , as $p_i \cdot (r_{reward} + \epsilon - r_{participate} - \text{EEV}(tx_V)) > p_i \cdot r_{reward}$. As every economically rational miner would pick this strategy, a fine-grained distinction between rewarding/compensation schemes for Π' is irrelevant in this case.

To analyze the profitability in the case where the victim and/or altruistic miners have some hashrate ($p_A + p_V > 0$), distinguishing between different rewarding/compensation schemes becomes relevant. For *effort-related compensation*, the main aspect regarding the profitability of the attack for a NC miner is that transactions tx_V , which has to be excluded, cannot provide extractable value, e.g., transaction fees, to any Π' compliant miner. Due to the compensation scheme, bribes do not lose any block rewards even if they support a later unsuccessful attack. Therefore, the situation for individual economically rational miners becomes the same as if all miners would be economically rational. This means if $p_i \cdot (r_{reward} + \epsilon - \text{cost}_{participate} - \text{EEV}(tx_V)) > p_i \cdot r_{reward}$ economically rational miners will decide to participate in the attack and accept the bribe. If not all miners participate in the attack, then the probability of finding a block on the attack chain is even better. As all contributed blocks on the (longest) attack chain get compensated, this situation might even be more attractive for potential bribes.

For Eve, initiating an EXCLUSIONGAME is profitable if the extractable value from excluding the transaction is higher than the costs for bribing all miners between n and $n + \ell$ and the operation of Π' . Therefore the expected extractable value of the chain c' , which does not include tx_V has to be larger than the promised bribes, $\text{EEV}(c') > \ell \cdot \epsilon$. Moreover, the attack requires a budget of $\ell \cdot \max(r_{reward}, \epsilon)$ to plausibly ensure that all payouts can be performed, even if the attack fails.

For details regarding success probability and costs of such attacks, see [JSZ⁺19, JSSW22b], or the Chapters 4 and 5 of this thesis.

7.6.6 Countermeasures

If the victim recognizes the attack, a countermeasure would be to raise the available fee for tx_V such that $\epsilon - (r_{participate} + \text{EEV}(tx_V)) \leq 0$. If miners are facing participation costs ($r_{participate} > 0$), or the attacker is facing operational costs $r_{operational} > 0$, this defense is inherently cheaper than the attack. This makes raising the fee an efficient counter-attack.

If Assumption 8 (independent transactions) does not hold, then all potential transactions which will build up on t_z have to be considered as well. This can be modeled by increasing $\text{EEV}(tx_V)$ by the extractable values of these potential transactions, which could not be included if tx_V has not been included.

If Assumption 9 (no payment transactions to miners) would not hold, then tx_V or any subsequent transaction might contain a payment to a player P which is also a miner $P \in \mathcal{M}$. Therefore, P would face a direct loss if the attack is successful. In this case, the incentives of the respective miner change, and he is no longer indifferent ($P \notin \mathcal{I}$).

Instead, he becomes a victim ($P \in \mathcal{V}$) and will work against the attack with his hashrate if he is rational ($P \in \mathcal{R}$). The same would hold true if P is altruistic ($P \in \mathcal{A}$) as he would want to stick to the protocol and include tx_V . Therefore the cases $P \in \mathcal{A}$ and $P \in \mathcal{R}$ would lead to the same behavior in this situation. The only difference would be that an altruistic miner would not continue mining on a fork that is behind the longest chain, while this would be an option for a rational miner if the stakes are high enough.

For an analysis of counter-attacks where the victims do not give up that easily, see [JSSW22b], or Chapter 5 of this thesis.

7.7 Adversarial Game to Revise Transactions (deep-fork)

For this scenario, we assume an attacker Eve (E) that wants to manipulate the outcome of a sub-game of the infinite NC game to her advantage, i.e., perform a *deep-fork*. Let τ be the original sequence of transactions containing some transaction tx_E which represents a payment to some merchant V . Let c_{n+k_V} be the original chain of blocks, including τ . This chain occurred during normal operation of the NC game.

$$\tau = (tx_0, tx_1, \dots, tx_E, \dots) \quad (7.63)$$

$$c_{n+k_V} = (b_1(tx_0), \dots, b_{n+1}(\dots, tx_E, \dots), \dots, b_{n+k_V}(\dots)) \quad (7.64)$$

Now let's assume Eve has a preferred sequence in which she wants to change tx_E to a different transaction tx'_E because she wants to launch a double-spend. Since NC does not offer finality, the original sequence τ is subject to change, and thus the payout tx_E can be changed to tx'_E in retrospect. The resulting desired transaction sequence τ' can be equivalent to τ except for transaction tx_E .

$$\tau' = (tx_0, tx_1, \dots, tx'_E, \dots) \quad (7.65)$$

$$c'_{n+k_V+1} = (b_1(tx_0), \dots, b'_{n+1}(\dots, tx'_E, \dots), \dots, b'_{n+k_V+1}(\dots)) \quad (7.66)$$

The resulting chain c' must have at least one more block s.t. it is longer than c_{n+k_V} .

The goal of the attack is it to convince economically rational miners of the NC game to mine the sequence c' as soon as block c_{n+k_V} is reached. Therefore, Eve constructs an adversarial game Π' that credibly promises rewards to influence the overall payouts of a specific sub-game of the NC game, that led to sequence c_{n+k_V} . If successful, Π' incentivizes the creation of the chain c' as a result. This chain then can be replayed/submitted to the original NC game and thereby also create a new longest chain confirming tx'_E .

7.7.1 Assumptions of the AG ExclusionGame

The assumptions from Section 7.7.1 still apply. Moreover, as we are double-spending a payment here, we have one additional assumption in this case

Assumption 10 (External Transaction Utility). Each time a payout or payment is considered confirmed after $k_{recipient}$ blocks, this creates an external utility, e.g., a (digital)good is received in return for the payment. This requires that there are at least two resources in the individual set of resources \mathcal{R} of every participant.

Note that assumption 10 also holds for payout transactions as they can also be used by miners to perform payments as the recipient can freely be chosen by them.

7.7.2 Rules of the AG RevisionGame

For the attack, Eve (E) creates a new adversarial game Π' called REVISIONGAME, that is defined by the 8-tuple $\Pi' = \langle \Pi, b_n, \text{EEXECUTE}(\cdot), \text{EVALID}(\cdot), \epsilon, k_E, \ell, k_V \rangle$.

Hereby, the arguments to a game Π' initiated by E are: The specification of the original/targeted cryptocurrency, that itself is a NC game Π . This specification contains all necessary predicates, functions as well as the genesis block of Π . The current latest block b_n in the targeted game Π . This is the point in the chain to which the system's state is reverted when the attack starts. A new function $\text{EEXECUTE}(\cdot)$, that only produces a response that is not \perp if a desired state is computed in the required block interval between n and $n + k_V$. This is only the case if the effects of tx'_E have occurred in the respective block interval. For all other inputs, the response messages from $\text{EXECUTE}(\cdot)$ and $\text{EEXECUTE}(\cdot)$ are equivalent. The computed states of $\text{EXECUTE}(\cdot)$ and $\text{EEXECUTE}(\cdot)$ are the same for all inputs. Moreover, such as $\text{EXECUTE}(\cdot)$, the function $\text{EEXECUTE}(\cdot)$ is assumed to run in polynomial time. As a wrapper around the new execute function, a function $\text{EVALID}(\cdot)$ is used, which works the same as the original function $\text{VALID}(\cdot)$ but invokes $\text{EEXECUTE}(\cdot)$ instead of $\text{EXECUTE}(\cdot)$.

The offered bribe per block is denoted by ϵ , and the fork depth of the attack is given by k_V and thus is defined indirectly by the victim of the double-spend. The parameter k_E denotes the security parameter of the attacker Eve and represents the waiting period after which the submitted attack sequence is considered confirmed by Π' . The duration of the attack is dependent on the progress of the individual chains and the overall budget of Eve to finance the attack. We denote the overall number of blocks for which bribes could be financed by N . In the attack initialization itself, Eve publishes bribing funds for $\ell < N$ blocks. Thereby she has the possibility to extend the attack further if she deems it has a realistic chance of success. Between a block b_{n+k_V} and a block $b_{n+k_V+N+k_E}$, there is a block at height $n + k_V + \ell$ until the attack, and thus the race of both chains is currently funded. In the best case, only one additional block in c' is required ($\ell = 1$) as all miners are economically rational and switch to the attack chain immediately.

Of course, the structure of the adversarial game has to plausibly assure that the offered reward ϵ is indeed paid out if Π' terminates. This includes a guarantee that ℓ (and its associated funds) can only be increased and not reduced. As Eve is not in possession of infinite funds, the attack will run out of funds eventually - and thus terminate. Depending on the concrete implementation, the payment can happen in-band (i.e., in the targeted

cryptocurrency Π), or out-of-band (i.e., in a different cryptocurrency). In both cases, this might be ensured with the help of a smart contract, for example, demonstrated in [JSZ⁺19]. Depending on the concrete rewarding scheme of Π' and the expressiveness of the available smart contract languages, either in-band or out-of-band payments are easier to implement (see [JSZ⁺19] for details).

The adversarial game Π' has two termination conditions. The outcome of Π' is determined by the condition which happens first.

Termination condition I (success)

A player of Π' has to provide the adversarial game with a valid input sequence c' of length at least $n + k_V + \ell + k_E$, where the following conditions hold:

$$b_n \prec c' \wedge \quad (7.67)$$

$$\text{LENGTH}(c') \geq n + k_V + \ell + k_E \wedge \quad (7.68)$$

$$\text{VALID}(c') \wedge \quad (7.69)$$

$$\text{LENGTH}(c'_{n+1}) = n + 1 \wedge \quad (7.70)$$

$$\text{EVALID}(c'_{n+1}) \quad (7.71)$$

In other words, the submitted blockchain c' confirms a sub-chain $c'_{n+k_V+\ell}$ from block n to block $n + k_V + \ell$, which does contain the effects of transaction tx'_E , i.e., $\text{EVALID}(c'_{n+1}) = \text{TRUE}$. If this is the case, then the game rewards every miner from block $n + 1$ to block $n + k_V + \ell$, with an additional ϵ per block in the new game Π' . Moreover, to provide the same incentive for *all* miners, including the ones who have already contributed blocks to the original chain c_{n+k_V} containing tx_E to mine on an attack chain, those original miners have to be compensated for their lost stale blocks from c_{n+k_V} as well.

Therefore, in the best case, the total amount paid out for blocks in the range $[n + 1, n + k_V + \ell]$ is:

$$\sum_{i=n+1}^{n+k_V+\ell} \epsilon + \sum_{i=n+1}^{n+k_V} r_{\text{reward}}. \quad (7.72)$$

After the rewards have been paid, the game terminates.

If the original chain also progresses, while the attack chain progresses, it is necessary to incentivize (i.e., fund) further blocks on the attack chain. For every additional block on the attack chain, an additional bribe ϵ has to be paid until the budget of the attacker (of N blocks) has been consumed.

If it is not possible to implement a function $\text{EVALID}(\cdot)$ with the available in-band or the desired out-of-band payment option, e.g., due to constraints of the respective smart contract language, specifying the desired sequence of transactions is an alternative. In this case, the attacker predefines the entire transaction sequence τ' , i.e., the content of

the blocks from $n + 1$ to $n + k_V + \ell$, as proposed in [JSZ⁺19]. In this case, the game Π' has to compensate the respective miners for missing rewards in Π , as the recipient of the payout transaction cannot readily be predetermined by the attacker. Therefore, also the regular block rewards r_{reward} have to be paid by Π' .

Termination condition II (fail)

A player of Π' provides the adversarial game with any other valid input sequence c of length at least $n + k_V + \ell + k_E$, where the following holds:

$$b_n \prec c \wedge \quad (7.73)$$

$$\text{LENGTH}(c) \geq n + k_V + \ell + k_E \wedge \quad (7.74)$$

$$\text{VALID}(c) \wedge \quad (7.75)$$

$$\text{LENGTH}(c_{n+1}) = n + 1 \wedge \quad (7.76)$$

$$\text{EVALID}(c_{n+1}) = \text{FALSE} \quad (7.77)$$

In other words, a different blockchain c which does not satisfy $\text{EVALID}(c_{n+1})$, has been submitted before a desired chain c' has been submitted. Note that such a chain c can also be submitted to Π' by Eve if she observed this chain in Π and the AG has not yet terminated, i.e., if this is the first blockchain of length at least $n + k_V + \ell + k_E$. Again also, for this AG, several different rewarding schemes can be imagined. In the following, we will describe two, namely *no compensation* and *effort-related compensation*.

No Compensation: In this case, nothing is paid if the attack is unsuccessful. The history revision contract proposed in [MHM18] is comparable to this scenario. In this attack, the in-band bribe is only paid if the required state has been reached. The condition if the required state has been reached can be checked by a smart contract that is in possession of the required funds. The drawback of this approach is that the risk of the attack being successful is on the side of the participating miners. This can be changed in the effort-related compensation scheme.

Effort-Related Compensation: In this case, the risk of the attack being successful is shifted to the side of the attacker. Therefore, miners are more likely to participate since they do not lose anything if the attack is unsuccessful but might gain higher profits if the attack is successful. A practical example are the pay-to-win attacks described in [JSZ⁺19].

To correctly assess the contribution of each miner to the attack and at the same time prevent abuse, the adversarial game has to maintain multiple candidate chains. This means it extends one, or multiple chains $c_{candidate}$, which are initialized with b_n . A block

b_{new} is appended to $c_{candidate}$ if the following conditions hold:

$$\text{PREV}(b_{new}) = \text{HEAD}(c_{candidate}) \wedge \quad (7.78)$$

$$b_n \prec (c_{candidate}, b_{new}) \wedge \quad (7.79)$$

$$n < \text{LENGTH}\left((c_{candidate}, b_{new})\right) \leq n + k_V + \ell + k_E \wedge \quad (7.80)$$

$$\text{VALID}\left((c_{candidate}, b_{new})\right) \wedge \quad (7.81)$$

$$(7.82)$$

If now the first candidate chain reaches a length of $n + k_V + \ell + k_E$, the respective termination conditions of Π' are checked: If Π' terminated successfully, the miners from block $n + 1$ to block $n + k_V + \ell$ of the respective chain can collect their additional bribes ϵ and miner of now stale blocks from b_{n+1} to b_{n+k_V} get a refund of their lost block rewards of r_{reward} per block.

If Π' terminated unsuccessfully, the miners from block $n + 1$ to block $n + k_V$ of the largest candidate chain $c'_{candidate}$ where $b'_{n+1} \in c'_{candidate} \mid \text{EVALID}(b'_{n+1}) = \text{TRUE}$ can collect a refund for their contributed blocks in the range $n + 1$ to $n + k_V + \ell$ in the height of the respective block reward r_{reward} . This ensures that blocks of complacent miners are compensated, even if the attack is unsuccessful and c' did not become part of the canonical chain. Therefore, the total amount of compensations that have to be paid if the attack fails is:

$$\sum_{i=n+1}^{n+k_V+\ell} r_{reward} \quad (7.83)$$

Whereas the total amount of bribes and compensations paid if the attack succeeds is:

$$\sum_{i=n+1}^{n+k_V+\ell} \epsilon + \sum_{i=n+1}^{n+k_V} r_{reward}. \quad (7.84)$$

7.7.3 Practical Considerations of the RevisionGame

Depending on the variant of the attack and the targeted cryptocurrency, it can be implemented using in-band payments or out-of-band payments.

For the *no compensation* case, any cryptocurrency in which the addresses of miners are publicly known, and payments can depend on previous payments, the creation of mutually exclusive transactions is possible. Therefore, it is possible to pay a merchant V in one branch, and in another branch, the same coins/funds can be used to share profits with miners to incentivize them to contribute to the attack. Hereby, different methods are possible. One example would be the usage of direct payments to addresses belonging to miners that mined the blocks from n to $n + k_V$ on the original chain, as well as all the issuance bribing transactions to all (or a significant fraction of) miners for supporting the attacker desired branch. If the respective miners observe these transactions carrying

large enough funds, they could try to make the necessary branch part of the canonical chain to collect those funds. This would represent a basic form of AG with in-band rewards. In our model, publishing such bribing transactions changes the set of reachable system states and thus unlocks the possibility for some miners to reach states which are more profitable for them. If the expected profit for miners is large enough, this could also incentivize changing the current system state in retrospect, i.e., perform a deep-fork.

One practical example would be one of the first bribing attacks on Bitcoin proposed by Bonneau et al. [Bon16], where such a scenario is described for Bitcoin using anyone can spend transaction. This approach has the drawback that Eve could propose other unconfirmed transactions rewarding different miners (or herself) at any point in time. Using smart contracts to lock and issue promised payments, either in-band or out-of-band, mitigates this risk and makes the attacks more trustworthy from the perspective of potential collaborating miners.

As REVISIONGAME requires exactly the same primitives as the EXCLUSIONGAME, the proofs from Section 7.6.3 regarding efficient verification and eventual termination equally apply. As with the ORDERGAME and the EXCLUSIONGAME, the proof of theorem 3 shows that it is possible to construct an out-of-band variant of the REVISIONGAME that runs in parallel to the targeted NC game. This requires that the out-of-band rewards for the AG can be credibly paid out and are available in a resource $R \in \mathcal{R}$ economically rational players of the targeted NC game care about. If this is the case, then this composition could interfere with the incentives of such players in the NC game. If the state and all messages of the targeted NC game are publicly observable, as it is the case with so-called *permissionless* cryptocurrencies, it is not feasible to prevent the creation of such external adversarial games as all the data to construct them is publicly available. The requirements and options regarding communication/synchrony requirements between those two games are the same as in the ORDERGAME.

The variant with *effort-related compensation* has the advantage that it is theoretically risk-less for participating miners, as rewards for contributed attack chain blocks are paid to bribees even if the overall attack fails. This circumstance allows Eve to reduce the size of the bribes and thus the overall costs of a successful attack, as described in [JSZ⁺19, JSSW22b], or the Chapters 4 and 5 of this thesis. Moreover, for collaborating miners, it could be even more profitable to join an attack if not all miners do so as well because then the chance is higher to contribute a block to the attack chain as less hashrate is concentrated on this chain. This could make such attacks even more attractive in the first place. For more information, we refer to [JSSW22b] or Chapter 5 of this thesis.

7.7.4 Profitability of the RevisionGame

For our initial analysis, the assumptions *independent transactions* (see Section 8) and *no payment transactions to miners* prevent that there are additional fees or payments to miners besides the usual block reward and average fee. This makes it easier to estimate the required expected extractable value in terms of normalized block rewards that are

needed to incentivize and fund such attacks. For a detailed analysis as well as simulations on the profitability of deep-fork attacks, we refer to [JSZ⁺19, JSSW22b].

The question of what is economically rational and how it is possible to factor in a drop in the respective exchange rates is discussed in [JSSW22a], or Chapter 6 of this thesis. In general, it can be said that, that the rewards collectible through mining only grow linear with the number of blocks [Bud18]. If the potential gain from the double-spend/deep-fork is large enough, from an economic perspective, an arbitrary number of blocks could be reverted.

7.7.5 Countermeasures

As with the other AIM attacks, a counter-attack by the victim(s), which increases the expected extractable value of blocks on the original chain, is an economically rational countermeasure of the victim. Nevertheless, the situation is not desirable from a user perspective, as it requires every merchant to monitor the chain and potentially diminish his income as he could be forced to bid against maliciously funded racing attack chains.

Discussion and Directions for Future Work

“The incentive *may* help encourage nodes to stay honest.”

– Satoshi Nakamoto [Nak08]

In this chapter, we recapitulate the results of this thesis and relate them to the game theory literature to discuss our results. We then present some directions for future work and provide some concluding remarks.

8.1 Results

In this thesis, we have provided the first systematization of bribing, Goldfinger, Pitchfork, front-running, and other related attacks targeting the incentives of cryptocurrency participants (see Chapter 3). Thereby, we exposed them as different instances of the same category of attacks which we termed algorithmic incentive manipulation (AIM). Algorithmic incentive manipulation attacks utilize the properties and technologies of cryptocurrencies and distributed systems to facilitate the collusion of players by implementing enforceable side payments. These, side payments, also called bribes, can be triggered automatically by providing cryptographic proofs about the state of the targeted cryptocurrency system. By classifying different attacks according to their requirements, reward mechanism as well as their capabilities and impact on the consensus layer of cryptocurrencies (i.e., as no-fork, near-fork, deep-fork etc.), we addressed research question RQ1 (see Section 1.4.1). Our systematization also highlights that this attack space is not yet exhaustively explored.

Some of the identified gaps have been addressed by proposing new attacks and improved analysis techniques. We described a new attack called *Pitchfork* (see Chapter 2) which

demonstrates that public permissionless PoW-based cryptocurrencies are vulnerable against the (malicious) interlinking of their protocol rules using merged mining. Moreover, we provide proof that such an attack can only be counter-attacked if the attacker hashrate is below $1/3$, or below $1/4$ depending on the side effects of the direct counter-attack (see Section 2.4.2). Moreover, we describe a new attack called *Pay-to-win* (P2W) that is capable of reverting, reordering or excluding any transaction from a targeted public permissionless cryptocurrency based on NC, provided that the respective attacker has a sufficient budget on a different funding cryptocurrency (see Chapter 4). Our analysis includes simulations as well as a fully functional proof-of-concept implementation of the attack showing its technical feasibility. These new attacks show that the landscape of incentive-related attacks has not yet been exhaustively explored. Thereby answering the initial research question RQ2 (see Section 1.4.2) in the negative.

To better analyze the proposed attacks and other related variants, we devised a practically oriented model to calculate the success probability as well as the profitability of finite AIM attacks (see Chapter 5). By considering finite chain races, already contributed blocks, as well as the incentives of economically rational victims, we addressed research question RQ3 (see Section 1.4.3). Moreover, for cases where the attacker is only one block behind and tries to catch-up infinitely long, we were able to prove that his profitability on the fork increases beyond his expected profit for staying on the original chain as soon as he has a hashrate greater than $1/\phi^2 \approx 38.2\%$. Thereby, providing an explanation for the value $\approx 38.2\%$, which was a result from previous numerical analysis approaches [TJS16, MHM18].

In Chapter 6 we address research question RQ4 (see Section 1.4.4) and relate the concepts of *miner extractable value* (MEV) (or expected extractable value as a more general term) and algorithmic incentive manipulation (AIM). Thereby, we show that the concept of AIM using bribes and the concept of extractable value (EEV) are related: A bribe can be viewed as yet another source of extractable value. Moreover, we highlight that the MEV of other participants cannot readily be estimated, especially if they are interested in more than one resource/cryptocurrency.

In Chapter 7 we address research question RQ5 (see Section 1.4.5) from a technical standpoint. Therefore, we devise a model of Nakamoto consensus-based state machine replication (SMR) and prove that adversarial games to attack the mechanism design of such permissionless cryptocurrencies can technically always be constructed, provided that the targeted system is efficiently verifiable and eventually makes progress. Thereby, we also expose the assumptions and conditions under which such AIM attacks are technically possible.

We now discuss these results with respect to research question RQ5 regarding the incentive compatibility of cryptocurrencies and relate them to the game theory literature.

8.2 Game Theory, AIM, and Cryptocurrencies

The field of game theory is much older than the first cryptocurrencies based on NC. As incentives undoubtedly play a role in cryptocurrencies, a natural question is how the identified issues with permissionless cryptocurrencies regarding AIM relate to established results in the field of game theory and its associated rich body of literature. The problem of collusion and side payments (bribes) between participants has long been described in game theory and is known to be the Achilles heel of mechanism design [GBI18]. For truthful revelation regarding the funding of a public project, Green and Laffont [GL79] showed in 1979 that there exists no efficient, incentive-compatible mechanism in the presence of coalitions and bribes. This result was later extended by Schummer [Sch00] to other settings, showing that truthful revelation and efficient allocation are incompatible when considering collusion and bribes.

To address these impossibility results, assumptions are introduced which constrain the capabilities of players to collude [CM12]. For example, the number of possible colluders is bounded, the colluders are unable to keep their cooperation secret, or they are not able to make side payments. The line of research focusing on group-strategyproofness for example, has produced promising results under the assumption that agents only collude if they directly profit from it and that there is no way of compensating/bribing each other to redistribute utility [GBI18].

Other researchers have argued that “*collusion is only too human, and to develop a realistic theory of human interactions it will be crucial to consider mechanisms and solution concepts resilient to collusion*” [CM12]. Therefore, another branch of research tries to bypass previous impossibility results by changing the system model such that coalitions are required to have dominant strategies and/or introduce the ability to punish players [CM12, GBI18]. However, there are system models and settings which remain a problem, as noted in 2018 by Gorokh et al. [GBI18]: “*For our second negative result, we consider a collective decision making problem, where agents must choose between one of two options, and monetary reparations are allowed. Here we show that not only is collusion resilience impossible, but in fact, any non-trivial constant-factor approximation of welfare is impossible under coalitional dominant strategies. Thus, in a sense, collective decision making is the worst case scenario for mechanism design in the face of collusion.*”

Given these results, the question arises, what is the appropriate game-theoretic system model for distributed systems implementing permissionless cryptocurrencies, and what are the best possible guarantees that can be achieved by proper mechanism design?

8.2.1 Incentive compatibility

The question if cryptocurrencies based on NC are *incentive compatible* and thus can be considered strongly stable has been issued early on in the research community [BMC⁺15].

“ Nakamoto originally argued that Bitcoin will remain stable as long as all miners follow their own economic incentives, a property called *incentive compatibility*. *Incentive compatibility* has never been formally defined in the context of Bitcoin or cryptocurrencies; its prevalence as a term likely stems from its intuitive appeal and marketing value. [...] In game-theoretic terms, if universal compliance were shown to be a Nash equilibrium, this would imply *incentive compatibility* for Bitcoin as no miner would have any incentive to unilaterally change strategy. This would imply a notion of *weak stability* if other equilibria exist and *strong stability* if universal compliance were the sole equilibrium. If on the other hand *non-compliant strategies* dominate compliance, we must ask whether the resulting strategy equilibrium leads to stability for the consensus protocol.” [BMC⁺15]

There are several different degrees and definitions of incentive compatibility. Informally, incentive compatibility describes a state in game theory and economics in which the incentives that motivate the actions of individual participants in a specific system are consistent with following the rules established by the group. This property is especially important in interactions in which a participant does not know perfectly what another participant knows or does. If the rules of the interactions are designed such that a participant with more information is motivated to act in the interest of the other party, or has less incentive to exploit an advantage, the result is incentive compatibility [Jam14].

There have been several indicators that incentive compatibility (or *group strategyproofness*) is too much to ask for in the context of cryptocurrencies based on NC (cf., [ES14, Bon16, Bon18, JSSW18, JSZ⁺19, FB19, KF19, JSZ⁺21b, FB19]). In [BMC⁺15], the authors divide the question regarding the incentive compatibility and, thus, the stability of NC into three sub-questions:

1. “*Stability with bitcoin-denominated utility*”, which assumes that the sole objective of a miner is obtaining the respective currency unit of the system.
2. “*Stability with external-denominated utility*”, where the miners have a utility function that allows them to convert their coins into other assets.
3. “*Stability with incentives other than mining income*”, where the miners’ utility is not purely derived from mining rewards but external resources. Goldfinger [KDF13] attacks and feather forking [sAM] have been used as examples.

The results provided in this thesis show that if *incentives other than mining income* (e.g., *external-denominated utility* such as other financial resources) are, in principle, able to change the behavior of miners in the targeted system and can be provided by an attacker in sufficient quantity, then adversarial games can be constructed for permissionless cryptocurrencies based on NC to reward attacker desired outcomes with the respective resources. In this thesis, we have shown that this holds for any targeted instance of NC that is efficiently verifiable and eventually makes progress. As a result,

resilience against such attacks cannot be ensured by purely technical means. Instead, economic bounds on the availability and/or attractiveness of external resources have to be assumed to provide security guarantees under such economic attacks. Therefore, we argue that the question of whether NC can be considered *stable under incentives other than mining income* or sufficient *external-denominated utility* has to be answered in the negative. Thereby, we also support the argument presented by Ford and Böhme [FB19] that rational participants can be turned Byzantine, given a sufficient incentive.

8.3 Future Work

As we have shown in this thesis, AIM can lead to interesting side effects and attacks in NC-based public permissionless cryptocurrencies. This encourages further research and raises new questions that have not yet been conclusively answered.

What is (economically) rational in the context of an AIM attack? All AIM attacks assume some form of (economically) rational behavior of participants. In practice, it is hard to define rational behavior in a general way, as also the individual investments and the long-term interests of miners (or stakers) play an important role. However, there may be scenarios where miners are capable of providing PoW or signatures for a targeted cryptocurrency, but at the same time, they do not have any long-term interest in the well-being of the target. Consider the real-world example of Bitcoin and BitcoinCash, or Ethereum and Ethereum Classic, which utilize (or utilized) the same form of PoW and can be considered rivals. Thus, the question if the proposed attacks are possible in practice is difficult to answer scientifically.

Large-scale temporary majority attacks, in which an attacker overtakes a cryptocurrency for a short period of time, have gained further practical importance as they have been observed more frequently in recent history. There is already empirical evidence from previous large-scale attacks by miners, especially on smaller cryptocurrencies [coi20, coi19b, coi19a, btg20], as well as AIM attacks that optimize extractable value through different types of front-running [DGK⁺20]. These cases demonstrate that large-scale attacks happen and that the topic of incentives in cryptocurrencies is an area that deserves further study.

How does the exchange rate of different cryptocurrencies relate to each other? Regarding the possibility of such attacks (especially Goldfinger attacks), the development of the exchange rates of different cryptocurrencies under large-scale attacks on individual systems is an interesting question. If the exchange rates of spared cryptocurrencies are not, or even positively affected, this could increase the probability of AIM attacks as evading negative economic consequences of attacks becomes a viable option.

What system model combinations for the targeted and the attacking system benefit or prevent attacks? The system model, especially the synchrony and finality

properties of the targeted system, influence the design of adversarial games aimed at attacking it. Adversarial games rely on input from the targeted system. Therefore, specific characteristics of the target system model propagate to the adversarial game or the attacking system, respectively. As mentioned in the discussion, certain combinations of system models seem to hamper attacks, while other combinations make the construction of attacks easier. Mapping the landscape of different combinations, as well as resulting meta-system models and their exact influence on the construction of attacks, is subject to future work.

How are permissionless PoS cryptocurrencies affected by AIM attacks? Since all considered attacks primarily target PoW cryptocurrencies, the applicability of AIM on PoS cryptocurrencies is not sufficiently understood yet. It remains to be understood which techniques are transferable to PoS cryptocurrencies and which additional mitigations (e.g., providing collateral, slashing) can increase the induced costs of attacks in this setting.

8.4 Conclusion

As Satoshi Nakamoto pointed out [Nak08]: “*the incentive **may** help encourage nodes to stay honest*”, but as shown in this thesis, incentives may as well encourage nodes to act maliciously or Byzantine.

We have shown that bribing, front-running, as well as Goldfinger attacks are no isolated attack examples. Instead, they belong to the larger category of algorithmic incentive manipulation attacks, which we described within this thesis. We have proven that it is always computationally possible to construct adversarial games that target permissionless cryptocurrencies (based on our model of NC), under the assumption that the targeted system is efficiently verifiable and eventually makes progress. Our results show that, even if the cryptocurrency internal transaction semantic is sufficiently restricted, adversarial games can always be constructed using out-of-band rewards that incentivize reaching the attacker’s desired system states. This includes arbitrary ordering- and exclusion of transactions, as well as deep-forks. The main conditions for this to be possible require that the respectively capable miners have perfect information regarding the adversarial game and that the promised out-of-band payments are sufficient and happen in an external resource that is deemed valuable for the respective miners. The practical technical feasibility of such attacks has been demonstrated by a proof-of-concept of an AIM attack on Bitcoin, which can be operated and funded from an Ethereum smart contract.

Given imperfect information regarding the available external resources and the attribution of transactions to players, it is impossible to accurately assess the economic security of prevalent permissionless cryptocurrencies based on NC, as the potential extractable value of specific actions for other participants is unknown. Moreover, capable actors could be incentivized to join for the sole purpose of attacking the system (which has already happened in some smaller cryptocurrencies [coi20, coi19b, coi19a, btg20]). Therefore, we

agree with the argument presented by Ford and Böhme [FB19] that economic rationality can lead to Byzantine behavior given sufficient incentives.

Even if no other external resource is available for out-of-band payments, depending on the expressiveness of transactions within a targeted cryptocurrency, AIM attacks utilizing in-band payments can also incentivize attacks with devastating effects. A key observation that was already noted by Bonneau [Bon16] and Budish [Bud18] is that the security guarantees of Nakamoto consensus against any bribing attack which facilitates deep- or near-forks are *linear* in the number of blocks in terms of the financial resources required. In contrast, the achievable security guarantees of many other investments in IT security, for example, in cryptography, are designed to “*yield convex returns*” [Bud18].

Without an honest majority assumption, the security guarantees not only depend on computer science aspects (e.g., the cryptographic primitives) but also on economic considerations, and the assumed behavior of players. Therefore the security of permissionless cryptocurrencies lies at the intersection of economics, game theory, mechanism design and computer science. It is well known that the answer to the question “is it secure?” always depends on the chosen threat model and that there is no 100% secure system in practice. Therefore, on the positive side, algorithmic incentive manipulation through adversarial games provides a price estimate for the achievable level of economic security. So although we have shown that it is impossible to implement solely technical protections against some AIM attacks on permissionless cryptocurrencies based on NC, it is still possible that there are countermeasures that increase the costs and thus reduce the likelihood of such attacks. For example, consensus algorithms with finality and order fairness, reward distribution over multiple blocks as well any long-term commitment of nodes in the form of collateral in combination with punishment/slashing for malicious behavior could drastically increase the costs for successful AIM attacks. Further research is required to identify the most promising approaches.

Given the environmental impact, one of the most critical ongoing changes is replacing NC based on PoW with different consensus systems. In this regard, different NC-style Proof-of-Stake (PoS) systems have established themselves as potential candidates. In any case, a good understanding of the achievable security guarantees of the consensus layer is of utmost importance for all applications that build on top of such systems, such as smart contracts and payment channels. Therefore, a good understanding of algorithmic incentive manipulation is crucial, especially for PoS systems, as in such systems, the distribution of funds is directly related to the security of the underlying consensus layer. With this thesis, we lay the groundwork for a better understanding of algorithmic incentive manipulation in distributed systems and the resulting security implications.

Appendices

A.1 Merged Mining Security Considerations

In this section, we briefly outline some practically relevant security considerations regarding merged mining that have, to the best of our knowledge, not been explicitly documented before. These remarks should serve as a starting point for the interested reader that might want to dig into the respective subjects.

A.1.1 Merged Mining with Simultaneous Direct Mining

In the early days of merged mining, Namecoin allowed merge mined blocks with Bitcoin, as well as blocks that have been directly mined for Namecoin. Due to the fact that Namecoin was the first fork of Bitcoin, it shared the same code base. Thus large parts of the functionality, as well as the data structures, were similar to Bitcoin. This included the structure of the block header. In the way that merged mining was, and partially still is, implemented for these two cryptocurrencies, the cryptographic link to the auxiliary (i.e., child) block header of Namecoin is stored in the Bitcoin block as part of a Merkle tree. The Merkle tree root hash itself is stored in the coinbase transaction of the respective Bitcoin block.

Due to the close resemblance of a Bitcoin- and a Namecoin block header, it would have been possible to submit a Namecoin block header to Namecoin as a Bitcoin header for merged mining, provided that the required difficulty of the PoW has been achieved. This would have allowed an adversary to mine on a Namecoin block header directly, as well as to include a Merkle tree root hash in the coinbase of the associated transactions within the same Namecoin block. This Merkle tree root hash could have contained a cryptographic link to another Namecoin block for an alternative Namecoin chain. With this construction, it would have been possible for an adversarial miner to mine on two different chains simultaneously. This, of course, is problematic in Nakamoto consensus,

as the miner can then later reveal a presumably “merge mined” Namecoin child chain that might be more favourable for the attacker e.g., includes a double-spend.

To the best of our knowledge, this issue was never explicitly mentioned for such designs. Luckily, this specific flaw was first prevented by some validation rule not directly focused on that specific subject. Later this flaw was eliminated by disallowing direct mining of most merged mined cryptocurrencies while only allowing merge mined blocks.

A.1.2 No Cryptographic Links Between Parent Chain Blocks

It is common in merged mining not to require cryptographic links between the parent chain blocks. The only requirement is that parent chain blocks include a sufficiently difficult PoW. The underlying assumption is that an economically rational miner would always want to extend the longest (heaviest) parent chain to potentially receive the rewards associated with the respective parent chain block.

Although, in some scenarios, the missing cryptographic reference between parent chain blocks might be problematic. For example, in a pay-to-win (P2W) attack, as described in Chapter 4, on a merge mined child cryptocurrency, the collaborating bribees would be able to simultaneously mine on the entire attack chain for the child cryptocurrency as the content is already predetermined by the attack smart contract. Thereby, each bribed miner can mine on a different block, while at the same time keep on mining for the parent chain as normal. If, in such a scenario, a bribed miner finds a valid PoW, he submits it to the attack smart contract and continues with another block. Suppose the attack(er) is able to assign miners to exclusively mine on specific blocks. In that case, no computation power is wasted in forks or repeated invalid brute force attempts by different miners.

Another scenario where this circumstance might be problematic is for cryptocurrencies that are merge mined with multiple parent chains, especially when miners of different parent chains collude and agree on block content and assigned slots with child chain blocks to mine on.

More generally, this means that the requirements of a *typical execution*, namely that no *insertion* or *prediction* [GKL20] occurred, do not hold in such merge mined cryptocurrencies in which no links between parent block headers are required. Thus the respective security proofs in [GKL20] are not directly applicable to such merge mined cryptocurrencies.

A.2 Optimal Transaction Ordering

We hereby want to analyze how hard it is to find an *optimal ordering* in this setting, where the player does not contribute her own transactions, i.e., $\mathcal{T}_i^i = \emptyset$. Hence we call this approach *passive*, as no new transactions are created.

In comparison to Ethereum, the fees in Bitcoin are static i.e., they do not depend on the position of a transaction. Therefore, when constructing a block in Bitcoin, it is only relevant that no transactions in the new block are contradictory. If all Bitcoin transactions would carry an equally large fee, mining would be an instance of the *maximum independent-set problem*. In practice, Bitcoin transactions can have different (but static) fees, as well as different sizes. Therefore, the optimization problem a Bitcoin miner is facing is an instance of the *knapsack problem*, which is NP-hard ¹.

In Ethereum, the situation is even more complex as the fee (i.e., the used gas) can be dependent on the position of the transaction within the block. Depending on the position in a block, the transaction executes on a different global state and thus might require less or more instructions to execute. Therefore, it can consume less or more gas. So by optimizing the order in which transactions are included in a block, miners might be able to extract a higher amount of fees. This, for example, is the case when more gas is consumed by transactions. If we make the simplifying assumption that the number of transactions in a block is constant and that there are finitely many new transactions in the transaction pool $\mathcal{T} := \mathcal{T}_i \setminus \text{TXSEQ}(\text{CHAIN}(b_n))$, which have not yet been included in a block, then the upper bound for finding the optimal ordering for a new block template \bar{b}_{n+1} can be derived as follows:

Proposition 2 (Optimal ordering for new transactions). Finding the optimal ordering of a set of transactions for a block template requires, at most $\aleph! \cdot [|\mathcal{T}|]^\aleph$ steps if $|\mathcal{T}| > \aleph$ holds and $|\mathcal{T}|!$ steps if $|\mathcal{T}| \leq \aleph$ holds. Both scenarios require constant space, which is approximately \aleph .

Optimal ordering for new transactions. The number of possible combinations of size \aleph of all transactions in the set \mathcal{T} is given by $[|\mathcal{T}|]^\aleph$. For each of these combinations, the miner has to check all combinations of transactions to fill the block, which are $\aleph!$. If $|\mathcal{T}| \leq \aleph$ all transactions fit in the current block thus the number of possible combinations reduces to $|\mathcal{T}|!$. To select the best one, it is sufficient to only store the currently best candidate and replace it if better ordering has been found. This leads to approximately \aleph required space for one particular ordering configuration. \square

These bounds of naïve order optimization show that finding an optimal ordering starts getting computationally infeasible rather quickly for practical block sizes. For example, a full Ethereum block currently contains around 250 transactions. Under the assumption that the transaction pool would be equally large, this would already amount to $250! \approx 3.23 \dots \cdot 10^{492}$, which is far beyond being computable. Moreover, we have not yet considered the possibility that the miner can create transactions as well and add them to the transaction pool to increase the extractable value from other transactions already in the pool.

¹Cf. <https://freedom-to-tinker.com/2014/10/27/bitcoin-mining-is-np-hard/>

Of course there are good approximations for finding a profitable block, this is done, for example, in front-running and related approaches [DGK⁺20, QZG22, ZQC⁺21, ZQT⁺20, TIGS21]. Nevertheless, the overall complexity of finding the optimal sequence of transactions could be exploited by a briber who offers a block candidate \bar{b}_{n+1} defined by himself, which promises a certain (above average) reward to the respective miner (e.g., as described in [JSZ⁺19], or Chapter 4 of this thesis). Mining this block candidate provided by an adversary could be easier for the miner than investing the resources to search for a more profitable order on his own. This trend towards separating block content creation from block confirmation/mining is also something we see currently in designs for proposer builder separation (PBS) in Ethereum [But21].

A.3 Types of Extraction Optimizations Methods

In this section we depict different attacks as methods to optimize the extraction of value. Depending on the willingness to interfere with the rules of the protocol and the ability to create transactions we classify known attacks in this area. What all approaches have in common is that, from the perspective of AIM, they all rely on economically rational behavior of actors. For our classification we build upon previous classifications [EMC19, QZG22, JSZ⁺21b].

On a high level we divide value extraction optimization techniques into *unintrusive* and *intrusive* methods. Unintrusive methods do not interfere with the process of achieving consensus, i.e., require *no-forks* and thus have also been termed *no-fork attacks* [JSZ⁺21b]. Intrusive methods require interference with the process of achieving consensus, i.e., require a fork. Moreover, intrusive methods can be further divided into *protocol coherent* and *protocol changing* extraction optimizations. Their difference being that the latter changes the protocol rules and thus requires a so-called *hard-fork*.

Both, unintrusive and intrusive, variants can be incentivized through AIM attacks using in-band, or out-of-band payments/bribes. If out-of-band payments and bribes, issued in different cryptocurrencies, are considered, this implies that at least two cryptocurrencies are executed in parallel. In such a case, we have to differentiate between a *target* cryptocurrency in which the desired effects have to be triggered and a *funding* cryptocurrency in which (additional) rewards are issued conditioned on the actions taken in the target cryptocurrency.

A.3.1 Unintrusive methods

Unintrusive methods do not interfere with the process of achieving consensus, i.e., require *no-forks* and thus have also been termed *no-fork attacks* [JSZ⁺21b]. Some of these attacks can therefore be executed by miners as well as users. Unintrusive methods can be further separated into:

Passive no-fork attack, approaches which do not require the creation of transactions, but do require voting power, i.e., can only be executed by proposers. Examples are:

- *Passive no-fork order optimization* [DGK⁺20], in which the ordering which provides the highest value (e.g., in terms of fees) is selected without creating new transactions. Another possibility, would be to intentionally order transactions such that they consume the most gas in Ethereum. For an analysis on how to find the optimal ordering see Appendix A.2.
- *No-fork exclusion attacks* [MHM18, JSZ⁺19, WHF19], in which a transaction is not included into a block because there is a revenue opportunity for the miner in this case, e.g., due to a side payment (bribe or AIM attack), or because he himself is then able to profit directly e.g., by winning an auction. Since we are still in a *no-fork* scenario, the miner can only prevent or ensure the inclusion of tx if she mines the next block. If she wants to prevent the inclusion of tx for multiple blocks, the same miner has to mine multiple blocks in a row. An analysis of this case can be found in Appendix A.6.

Active no-fork attack, approaches which do require the creation of additional transactions. Examples are:

- *Active no-fork order optimization* [DGK⁺20], in which not only the ordering which provides the highest value in terms of fees is selected, but also additional transactions can be created if needed to extract additional value, e.g., a guaranteed profit opportunity through arbitrage is observed, or an *any-on-can-spend* output is observed and collected. This case includes all kinds of attacks such as *front-running* [EMC19, DGK⁺20], *back-running*, *sandwich-attacks*, etc. [QZG22, ZQC⁺21, ZQT⁺20, TIGS21].

Actors in this scenario are not necessarily required to be proposers (e.g., miners in context of PoW). As long as actors share some of their potential profits from the desired ordering with the respective miner, then both parties might be able to profit. This can be done by increasing the fee of transactions to incentivize their inclusion, as empirically observed by MEV bots [DGK⁺20, ZQC⁺21]. Alternatively, miners could be bribed directly to produce the desired ordering [JSZ⁺19, JSZ⁺21b]. Also, this behavior is already observed in current MEV extraction bots, which trigger code that directly bribes the beneficiary of the respective block [SJSW22].

- *Active no-fork exclusion attacks*, in which other transactions are excluded by broadcasting sufficiently many high priority transactions (e.g., with high fee) to fill all available space in blocks. An example would be the Fomo3D exploitation [fom18]. This approach has also been referred to as *clogging* [QZG22], or transaction *triggering* [JSZ⁺21b]. Attacks in this category can be executed by every actors that can create or incentivize sufficiently many transactions.

Alternatively, this can be achieved with a bribing contract that later offers a higher reward when a certain transaction was not included until a particular height. Examples of such attacks are described in [MHM18, JSZ⁺19, WHF19, KNW20], Chapter 4 and Appendix A.6 of this thesis.

A.3.2 Intrusive methods

which require interference with consensus i.e., a fork. Attacks in this category require the active participation of actors with voting power (i.e., miners with hashrate in context of PoW). We distinguish between *protocol coherent* and *protocol changing* methods. In a protocol coherent method an attacker preferred a certain branch regarding the resolution of a *temporary* fork. In a protocol changing method the rules of the preferred attacker branch are incompatible with the original rules and therefore necessitate a so-called *hard-fork*. Intrusive methods can be separated into the following categories.

Deep-fork attacks, where a fork with a depth of at least ℓ exceeding a security parameter k_V of some victim V is necessary (i.e., $\ell > k_V$). The victim defines k_V [GKL15, SZ16], and it refers to its required number of confirmation blocks for accepting transactions². In other words, the victim indirectly defines the required minimum fork length ℓ by his choice of k_V . The classic example of a deep-fork attack is the *double-spending attack* [Ros14, DKT⁺20], in which a transaction is revised, and the amount is transferred back to the original sender s.t., the sender can spend the same amount multiple times.

A double-spend necessarily requires the issuance of a transaction at some point in time. Therefore, the distinction between *active* and *passive* double-spending attacks makes no sense, as all double-spending attacks must be *active* attacks as there initially must be a transaction to double-spend. Moreover, in classic double-spending attacks, another conflicting transaction tx' has to be issued and included instead of tx . Although, issuing a conflicting transaction is not necessary for a double-spend to be successful, as demonstrated by *opportunistic double-spending attacks* [SJSW22].

Near-fork attacks, where the required fork depth is *not* dependent on some security parameter k_V chosen by a concrete victim, but forks might be required by the performed extractable value optimization. In other words, the attacker defines a gap k_{gap} , that is chosen independently from any other security parameter k_V . It could be smaller than any k_V , or larger³. If a miner identified a large enough revenue opportunity, she could decide to launch a near-fork attack. For example, if the currently available set of transactions \mathcal{T}' would allow the miner to extract more profit in retrospect, she could either mine certain more profitable blocks herself or change the order of already confirmed transactions so that they are more profitable. This, for example, is the case in a *time bandit attack*, where previously issued and already included transactions can be reordered to gain a higher trading profit due to cheaper purchases and more valuable sales. Hereby, the gap would be defined by the required maximum block height x to make these trades possible again, s.t. $x \leq n$ and $k_{gap} = n - x$. All transactions from block x to n are then available again for inclusion, as they are a subset of all known transactions, $\text{TXSEQ}((b_x, \dots, b_n)) \subseteq \mathcal{T}'$.

²We emphasize that each transaction has a recipient (and thus a potential victim with an individual k_V), in practice, there is no global security parameter k , that holds for all transactions.

³The length of k_{gap} also depends on the attacker's resources and willingness to succeed (e.g., to exclude a certain block).

Examples of near-fork attacks are:

- *Undercutting attack* [CKWN16], in which the miner forks the latest block (or multiple blocks) to mine the included transactions himself.
- *Censorship* or (*near-fork*) *exclusion attack* [MHM18, JSZ⁺19, WHF19], which is conceptually the same as the no-fork exclusion attack, but in this case, the extractable value for excluding a transaction is large enough to incentivize a near-fork. An early variant of this has also been termed *feather forking* [BMC⁺15].
- *Time bandit attack* [DGK⁺20], in which the attacker reorders transactions that have already been included in a block in retrospect. Since the point in time when this reordering happens is independent of the individual security parameter (k_V) of the involved transaction, this type of attack can be categorized as a near-fork attack.
- *Selfish mining* [ES14, NKMS16, SSZ16]. Selfish mining is also a form of near-fork attack in which the additional extractable value comes in the form of a relative increase in rewards after the difficulty has been adjusted.

Hard-fork attacks, where the required fork changes the protocol rules and thus is incompatible with the original rules. This special case is therefore called *protocol changing*. Most formal protocol security analysis necessarily assumes that the protocol under investigation is static. This is a justifiable assumption, as in most cases, the maintenance and initial distribution of the protocol are controlled by a central, trusted party. In cryptocurrencies, the situation is slightly different. Practice shows that protocol changes are necessary and have happened several times in the past in different cryptocurrencies, e.g., in Bitcoin, for introducing segregated witness (SegWit). In Section 7.2, we revisit different types of protocol changes and discuss their relation and implication regarding the financial optimization of certain actors. An example Robin Hood attack is provided in Appendix A.4.

A.4 Adoption of Protocol Changes and the Robin Hood Attack

Which rules the majority of miners adopt depends on a variety of factors. The information regarding the new rules must be available to the miners, as well as the time (block height) at which these new rules become active. Moreover, the EEV under the new rules, compared to the EEV under the old rules plays a role. New rules Π' , imply a new execute function $\text{EXECUTE}'$ and thus also a new predicate VALID' and $\text{REACHABLE}'$ as well as a new function FUTURE' . An economically rational miner i , would stick to the original rules and ignore the new rules (starting at block b_n), when these rules do not lead to any block with higher expected extractable value.

$$\begin{aligned} & \forall b_{n+1} \in \text{FUTURE}(b_n, \mathcal{T}), \\ & \neg \exists b'_{n+1} \in \text{FUTURE}'(b_n, \mathcal{T}), \text{EEV}(b'_{n+1}) > \text{EEV}(b_{n+1}). \end{aligned} \quad (\text{A.1})$$

If this is not the case, this provides room for incentive manipulation. For example, for an attacker who bribes miners to follow his favored branch of a chain split. In the past, schemes have been developed that allow for the atomic swap of coins in case of a chain split [MHM17]. The motivation for this was a bet between two wealthy Bitcoin users (Roger Ver and Loaded) on whether or not some chain split will happen. Although this theoretically enables two parties to bet on the execution of a chain split, it was not presented as an attack to increase the likelihood of this event actually taking place. In the Paper [JSSW22a], as well as in Chapter 6, an example is given for a case where a permanent fork could be beneficial for users, namely if the exchange rate of both resulting cryptocurrencies is more than the original exchange rate the initial cryptocurrency. In this case, the amount of resources for users increased through the fork. Historical, forking events in cryptocurrencies lead to such situations, such as the dispute that motivated the split between Ethereum and Ethereum Classic (2016-06-28) where the exchange rate of Ethereum stayed between \$ 12 USD and \$ 13 USD and Ethereum Classic started with an exchange rate of around \$ 1.6 USD, which effectively increased the resources of all users holding that asset before the split⁴.

In the following section, we want to sketch an attack that is aimed at increasing the likelihood of a protocol change and, thus, a resulting chain split.

A.4.1 The Robin Hood Attack

We assume that there is a participant or set of participants \mathcal{V} which is known to possess a substantial amount of funds. Moreover, the associated addresses or unspent transaction outputs holding these funds are publicly known as well. Then some party can propose a changed set of rules Π' , which expropriates \mathcal{V} and redistributes these funds to all other mining participants. This would be an *overlapping and conflicting* change and requires that the new rules Π' are known to every miner. As long as a potential drop in the exchange rate is less than the gain from the distributed funds, it could be profitable for the majority of miners to accept this protocol change. If participants in \mathcal{V} do not control any hashrate, they cannot even intervene directly. In this case, they would be dependent on miners to support the original rules. Knowing this, supporting miners might require some financial compensation for doing so, leading to a loss of funds as well. If neither side gives up, the protocol change would turn into a permanent chain split and thus result in two different cryptocurrencies.

Given the distribution of coins in prevalent cryptocurrencies, substantial amounts can be attributed to certain parties. In Bitcoin, for example, addresses that are attributed

⁴cf. <https://www.investing.com/crypto/ethereum/historical-data> and <https://www.investing.com/crypto/ethereum-classic/historical-data>

to Satoshi Nakamoto hold around 1.1 million BTC, and addresses attributed to the company Binance hold around 200 thousand BTC, to just provide two examples ⁵.

A.5 Transaction Ordering attack (in-band)

This *no-fork* attack pays additional rewards within a targeted cryptocurrency (in-band) to miners for reordering *unconfirmed transactions*, comparable to front-running attacks [EMC19, DGK⁺20]. In front-running attacks, the adversary increases the chance of his transaction being included before others by increasing the transaction fee paid to miners. However, the result is an *all-pay auction*: even if the attack fails, the high-fee transaction can be included by miners. As such, the adversary must *always* pay the fee, independent of the attack outcome [DGK⁺20]. In contrast, our attack ensures the adversary pays colluding miners only if the attack was successful, i.e. if the desired transaction ordering was achieved.

A.5.1 Description

Initialization. The adversary (Blofeld) observes the P2P network and initiates the attack once he sees a victim’s (Vincent) transaction tx_V , which he wants to front-run (e.g., registering a domain name or interacting with an exchange). First, Blofeld publishes his front-running transaction tx_B . Simultaneously, he initializes an *attack contract* with the necessary information on the desired outcome, as well as the locked bribe, which serves as an incentive for miners to work towards reaching the attacker desired state. How this can be implemented on a technical level in Ethereum is discussed in Section A.5.2.

Attack. If the attack is successful, colluding miners generate a block that has the desired ordering of transactions. Note: even if the victim attempts to update the original transaction tx_V with tx'_V , e.g. using *replace by fee* [Wik], tx_V remains valid and can alternatively be included by miners to invalidate tx'_V . Rational miners will hence include tx_B and tx_V in the specified order, fulfilling the payout conditions, as long as this results in the highest reward.

Payout. After k_B blocks (k_B is the blockchain’s security parameter defined by the attacker in this case), miners can claim their payouts, whereby the smart contract first checks if the ordering of the two transactions is as specified.

A.5.2 Details and implementation of tx ordering in-band

On a high level, we differentiate between three methods to implement functions for verifying a certain transaction ordering in Ethereum. The first method only relies on proofs over the transaction trie of a given block to verify the desired transaction ordering. The second method tries to verify the desired state. The third method enables the attacker to specify the entire content of the desired block upfront. In the first two

⁵cf. <https://bitinfocharts.com/de/top-100-richest-bitcoin-addresses.html>

versions, the checks are performed at the level of individual transactions, whereas the third method verifies the state of an entire block.

Verify transaction ordering

For this technique, it is sufficient that identifiers of the two transactions, the desired order ($tx_B < tx_V$), the block in which the transaction(s) are to be included, and a bribe ϵ , are provided as a transaction to the attack contract. Thereby, the respective transactions, as well as the initialization transaction to the attack contract, can be included in the same block. Once the initialization transaction has been included in a block, (i) the configuration can no longer be changed, and (ii) the bribe is locked until the attack times out. This is necessary to prevent the attacker from attempting to defraud colluding miners by altering the payout conditions after the attack was executed.

The verification in this method works via a transaction trie inclusion proof provided to the attack smart contract. Since the key in the trie is the index of the transaction in the block and the value is the transaction hash, the ordering of any two or more transactions can be proven to a smart contract in retrospect.

The advantage of this approach is that it is conceptionally simple, but it bears certain drawbacks. Let us assume the transaction hash of the involved target transaction tx_V changes e.g., if a transaction was updated via *replace by fee*, or a completely different but conflicting transaction from the same address with the same nonce has been issued tx'_V . This case can still be captured by an attack contract which also checks the nonce of the respective transaction. Since the original transaction tx_V is still valid and can be included by a complacent rational miner, all transactions with the same nonce from the same account become invalid.

A problem arises if the victim publishes another transaction tx''_V , from a different account that has not been included in the initialization of the attack contract. This transaction might be semantically equivalent to tx_V , e.g., it would register the same name in sENS but would not be covered in the attack condition of the contract. Thus, a naive contract only working with transaction hashes and nonces of known transactions can be fooled by a victim to pay out bribes, although the attack was not successful because tx''_V has been included before tx_B and just tx_V has been included after tx_B .

Verify operation on specific state

This approach addresses the issue of interfering transactions mentioned in the previous section in two different ways.

Retrospective check It is proven to the attack contract in retrospect that it has successfully operated on the correct world-/smart contract state before any funds are unlocked.

Up to Ethereum EIP-150 revision, the *transaction receipt* also contained the *post-transaction*⁶ state R_σ . This would have allowed proving to the attack contract the state before any transaction as well as the state after a specific transaction. Unfortunately, the post-transaction state was removed from the transaction receipt for performance reasons.

A currently working generic method for Ethereum around this would be to require that the racing attack transaction has to be at index 0 in the new block mined by the miner. It would then be possible to prove to the attack contract in retrospect that the specified transaction at index 0 operated on a specific world state i.e., the world state of the previous block, e.g., where the name to register was not registered yet. The only way to also generically prove that the resulting state was indeed the required one without any side effects is that only transactions which are directly relevant to the attack are included in the new block in the respective order because then the resulting world state can be pre-computed. This, of course, renders the attack more expensive and less generic.

Runtime check During runtime, a smart contract in Ethereum does not know at which position the transaction that invoked the contract is located in the respective current block. Moreover, it is not possible to query the indices of other transactions during runtime. An alternative to working with indices of transactions is working directly with the required states. The attack contract checks if it is operating on the correct world state directly before even performing the attack e.g., check if the name it wants to register is available. If the attack contract encounters an error while performing an attack, it could prevent any future payouts of bribes.

In our front-running example, the front-running transaction can also be sent to the attack contract directly, which additionally works as a proxy or dispatcher and only forwards i.e., performs the transaction, iff a queryable attack condition is met i.e., the target contract is in a specific pre-defined state. Since the state (storage) of a contract cannot directly be accessed from another contract, only accessible functions, variables, and certain state variables such as `balance` can be accessed. Note that for publicly accessible variables, getter functions are created automatically. These runtime checks ensure that no payments happen if the attack is unsuccessful. Summarizing, checking for the desired state at runtime (if possible) can make attacks more efficient, and at the same time, more complex attack scenarios can be envisioned.

Specify the entire list of transactions

In this method, the attacker needs to specify and broadcast the entire ordering of the respective block, i.e., the root hashes of all transactions and the desired block height at which a block containing this specific list of transactions should be mined. The bribes' task is to create the entire block, including a valid PoW. To construct the desired block, any potential bribee requires all associated transactions under the attacker-provided root

⁶The according Ethereum yellowpaper describing this is still available at <http://gavwood.com/Paper.pdf> (accessed: 2019-05-04)

hash. This necessitates an additional communication channel where this data is made available by the attacker to all potential bribees.

Unfortunately, for the attacker, the specified transaction hash cannot include the initialization transaction itself, as it would be required to include a hash over itself. Therefore, the initialization transaction needs to be included in any subsequent block. Suppose this initialization transaction can be included within the next 256 blocks. In that case, the attack smart contract is able to verify if the specified block within the last 256 blocks indeed has the specified transaction root hash, given the associated block header of this block as input. The transaction root hash from the given header can then be compared against the desired one, and the hash of the header can be checked against the block hash at the desired height with the EVM opcode `blockhash (0x40)`.

In the given construction, miners can verify that the block desired by the attacker with the specified transactions is valid and therefore be sure to receive at least the regular block reward for such a block. Moreover, in the best case, they would also receive the additional bribe later. As this approach already requires an additional communication channel between the attacker and potential bribees, a variation of this attack is possible in which the attacker only sends out signed messages, including the desired transaction root hash as well as the desired block height. These messages can then be included in a transaction created by the bribee that wants to redeem his reward from the pre-funded attack contract as proof that the attacker indeed confirmed the creation of the respective block.

To provide an additional guarantee to the potential bribees, the attacker over-funds the attack contract up-front with locked currency units. Thereby, he would ensure that there still is sufficient money in the attack contract to pay the bribe, even if the attacker would issue multiple signed messages containing transaction root hashes and block heights.

A.5.3 Evaluation with Rational Miners Only ($p_{\mathcal{R}} = 1$)

First, we assume a scenario where all miners act rationally, i.e., are bribable. Miners are incentivized to collude with the adversary, as the contract guarantees a reward $\epsilon > 0$ in addition to regular mining. Participation in the attack does not require to mine on an alternative fork. Hence colluding miners face no additional risk that their blocks will be excluded from the main chain.

A.5.4 Evaluation with Altruistic Miners ($p_{\mathcal{R}} + p_{\mathcal{A}} = 1$)

In theory, this attack is possible with any hashrate of bribable miners $p_{\mathcal{R}} > 0$. However, the higher the hashrate, the higher the chances of success. If 2/3 of the hashrate is controlled by rational miners, the attack is expected to succeed in two out of three cases. We refer to Section A.6 in the Appendix for an analysis where rational miners are additionally incentivized to near-fork main chain blocks to successfully remove an undesired block from the chain.

A.5.5 Counter Mechanisms

For a list of general counter mechanisms, e.g., require blinded commitments up-front, which can be used to avoid such a vulnerability during the design of smart contracts, see [DGK⁺20, EMC19]. In this section, we focus on counter-bribing as a mitigation strategy. Therefore, we distinguish the counter-bribing based on the point in time where the counter-attack is performed.

Immediate Counter-bribing As long as the new block has not been mined, an effective countermeasure against this attack is to immediately perform counter-bribing through the same attack mechanism. Hereby, the attacker and the victim engage in an *English auction*, as only the winner pays the bribe, instead of the *all-pay-auction* observed in other front-running [DGK⁺20]. This defensive strategy assumes that Vincent actively monitors the P2P network and immediately becomes aware of the attack.

Delayed Counter-bribing If Vincent only has an SPV (Simple Payment Verification [Nak08]) wallet, he may only recognize the attack after a new block with the intended ordering of the attacker has already been mined. Since Vincent is not in possession of any hashrate, he cannot directly launch a counter-attack to fork the respective block. Thus, the costs for a successful counter-bribing attack have become much higher than the costs for the original attacker Blofeld. For an analysis of how much it costs to remove one block from the chain, we refer to [JSSW22b] or Chapter 5 of this thesis.

A.6 Transaction Exclusion (in-band)

The purpose of this near- or no-fork attack is it to exclude one or multiple unconfirmed transactions from their generated blocks. The targeted cryptocurrency is Ethereum. Thus the attacker is able to use smart contracts to offer in-band payments to incentivize compliant behavior. Such attacks can be profitable for an attacker attempting to falsely close an off-chain payment channel (i.e., publish an old/invalid state) but prevent the victim from executing the usual penalizing measures [PD16, MBKM19, DW15]. In this section, we outline the general functionality of the attack as well as discuss the technical details of how such attacks could be implemented in Ethereum.

A.6.1 Description

The non-inclusion of a specific transaction should be rewarded with an in-band bribe. There are multiple ways to prove the non-existence of a specific transaction (or its effects) to a smart contract in Ethereum, which we discuss in Section A.6.2. In general, the attack can be separated into three different phases.

Initialization. The attacker knows some transaction tx_V which he wants to prevent from getting into the main chain. He then initializes the attack contract at block e_1 , specifying the transaction and the duration N (in blocks) of the exclusion attack.

Attack. The attack contract will pay an extra ϵ for every block mined between block e_1 and e_N that (i) does not include transaction tx_V itself and (ii) does not extend any block that included transaction tx_V . If an altruistic miner decides to include tx_V in his block e_i ($i < N$), colluding miners must perform a near-fork, i.e., extend block e_{i-1} rather than e_i , if they wish to receive rewards.

Payout. Collaborating miners can claim payouts once k_B blocks have passed after the end of the attack, i.e., at a block $e_T \geq e_{N+k_B}$, where k_B is the security parameter defined by the attacker.

A.6.2 Details and implementation of tx exclusion in-band

The two crucial aspects of this attack are: i) Determine if the unwanted transaction tx_V was included, and if so, in which block ii) Correctly reward complacent miners. To collect the reward, a rational miner has to submit the block header he mined in the respective range to the attack contract. The attack contract then checks if this block indeed lies in the respective interval in the recent history of the chain. In Ethereum, the last 256 block hashes can be accessed from within a smart contract. Thereby the smart contract can verify if a submitted block header is indeed part of the recent history. From the submitted block header, the contract can also extract the beneficiary / coinbase address of the respective miner directly.

Transaction inclusion proof

Most PoW blockchains use accumulators, such as Merkle trees, to store and efficiently prove the inclusion of transactions in a block. However, proving the non-existence of an element in Merkle trees is inefficient. To this end, the attack contract will reward any submitted block between e_1 and e_N *unless the adversary submits an inclusion proof* for tx_V before the payouts are claimed in block e_T . If the adversary proves that a block e_x included tx_V , any blocks extending e_x , i.e., e_{x+1}, e_{x+2}, \dots , will not receive any payouts. Figure A.1 shows a failed attack where tx_V was included in block e_3 - thus, only blocks up to, but not including, e_3 are rewarded.

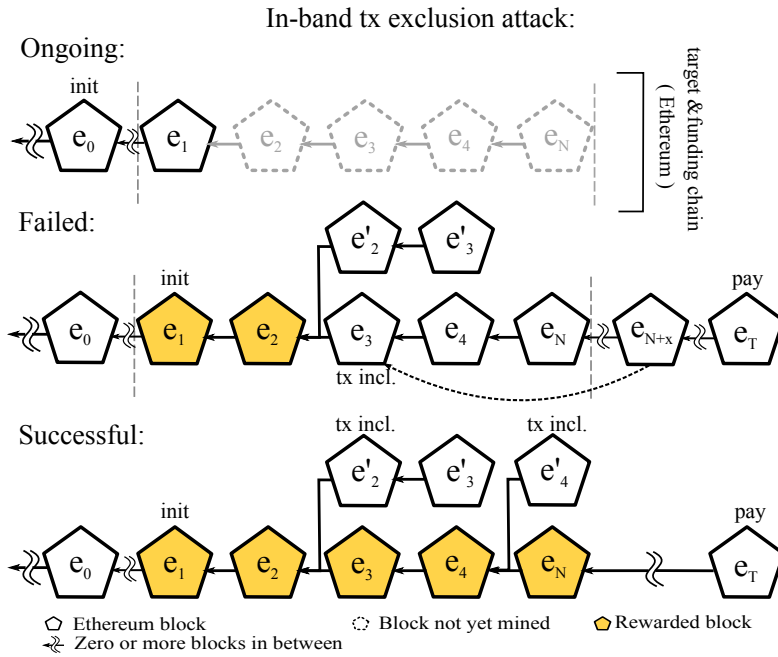


Figure A.1: The figure shows an ongoing-, a failed- as well as a successful Transaction exclusion attack with in-band payments. The attack is initialized when the attack contract is published in block e_1 . If the unwanted transaction has been included, this can be proven to the attack contract, as shown in the failure case in block e_{N+x} . The payouts are performed in block e_T . The colored blocks are rewarded by the attack contract with an additional ϵ .

The naïve way of determining if tx_V has been included in a block is to request a Merkle Patricia trie inclusion proof, as described in Section A.5.2, that the respective transaction is part of a given block header that lies in the defined interval. This approach has the drawback of not detecting other semantically equivalent transactions with a different hash.

A way around this in an in-band scenario on Ethereum is to define state conditions that must be met depending on the use case at hand. For example, a transaction can be submitted to the attack smart contract containing an inclusion proof of a transaction to a specific (unwanted) address as a payload. If this transaction happened in the specified block interval, this could be considered a proof that an unwanted interaction with the respective address has occurred. Therefore, no rewards will be paid by the attack smart contract to miners starting from the block that includes the unwanted transaction. Thereby, care has to be taken to account for transaction obfuscation via proxy contracts, which perform message calls on behalf of a transaction from an externally owned account. These cannot easily be proven to a contract since the respective transaction has to be evaluated on the EVM with the correct world-state. Thus, this variant is only error-free if the unwanted transaction has to come directly from an externally owned account, e.g.,

as required by certain Tokens⁷.

Therefore, the safest variant is to check whether the state change or condition that an unwanted transaction should have triggered has occurred. For example, if the balance of a contract has been raised/decreased or if a particular publicly accessible state variable has changed in an undesired way. If the attack contract can check this before performing any payouts, it is only possible to collect rewards if the requested condition has been fulfilled.

Block template in-band

Another way around the previously outlined problem of proving that an unwanted transaction has not occurred is to specify what transactions are allowed to take place. Interestingly, this is easier in an out-of-band scenario than in an in-band scenario since the attacker has to convincingly ensure the collaborating rational miners that they will receive their bribes while defining the content of all blocks in a way that can be proven to the attack smart contract. At the same time, the content of the blocks also has to define those blocks, which leads to a recursive dependency since the transaction to the attack contract cannot define itself because its hash is not known in advance. Conceptionally, the same techniques, such as sending out signed messages of transaction root hashes and block heights, as described in Section A.5.2 can also be used in this case.

A.6.3 Evaluation with Rational Miners Only ($p_{\mathcal{R}} = 1$)

Estimating the costs of such an attack in a scenario where all miners are rational ($p_{\mathcal{B}} = p_{\mathcal{A}} = 0$ and $p_{\mathcal{R}} = 1$) and have perfect information about the attack is trivial. In this case, it is a no-fork attack, and the respective transaction would not be included in the blockchain as long as the bribe ϵ for non-inclusion surpasses the fee miners can gain from including the respective transaction, i.e., $\epsilon > fee(tx_V)$.

A.6.4 Evaluation with Altruistic Miners

($p_{\mathcal{R}} + p_{\mathcal{A}} = 1$)

If a fraction of miners behaves altruistically, i.e., will not join the attack independent of profit, rational miners need an additional incentive to perform near-forks, excluding blocks containing tx_V .

Probability of success without a fork As rational miners find a block with probability $p_{\mathcal{R}}$, the likelihood of rational miners finding chains of consecutive blocks decreases exponentially in their length ℓ . For example, given $p_{\mathcal{R}} = \frac{2}{3}$ the probability of generating a chain of $\ell = 6$ consecutive blocks is merely 8.8%. However, another possible attack might only require delaying a specific reoccurring transaction at any point in time within

⁷Interestingly, a UTXO model would also be easier to censor if the output which has to be spent in an unwanted transaction is known.

the subsequent N blocks. For example, denying all transactions to a smart contract to prevent price updates. The probability for a miner with hashrate $p_{\mathcal{R}} = \frac{2}{3}$ to mine at least $\ell = 6$ consecutive blocks at least once within the next $n = 100$ total blocks is approximately 97.2%. This can be calculated for different values of n, ℓ and $p_{\mathcal{R}}$ by computing the matrix of the finite Markov chain depicted in A.2 with n as exponent, as shown in formula A.2.

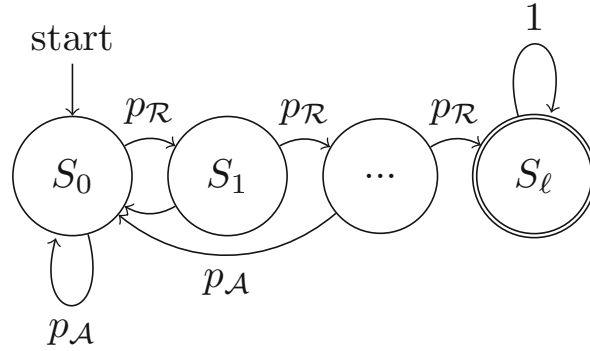


Figure A.2: Finite Markov chain for calculating the probability of mining at least ℓ consecutive blocks with hashrate $p_{\mathcal{R}}$.

$$P = \begin{bmatrix} p_A & p_{\mathcal{R}} & 0 & \cdots & 0 \\ p_A & 0 & p_{\mathcal{R}} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_A & 0 & 0 & \cdots & p_{\mathcal{R}} \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}^n \cdot \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix} \quad (\text{A.2})$$

Probability of success and costs with near-forks To increase the chance of success, the adversary must increase the bribe ϵ paid to colluding miners, to reimburse the risk of losing block rewards r_{reward} due to a failed fork. Assume altruistic miners mined a block containing tx_V . In this scenario, the *attack chain*, i.e., the fork produced by collaborating miners, which must not contain tx_V , is only one block behind the main chain. As such, the required bribing funds are significantly lower when compared to deep-fork bribing attacks. For an evaluation of this scenario, we refer to [JSSW22b] or chapter 5 of this thesis.

Comparison to Existing Attacks

A comparable attack allowing arbitrary transaction exclusion is `HistoryRevisionCon` [MHM18]. While `HistoryRevisionCon` only requires bribing amounts ϵ between $0.09375 \cdot r_e$ and

$1.4375 \cdot r_e$ (depending on how effective uncle block inclusion can be optimized), it also requires a substantial attacker hashrate ($p_B > \frac{1}{3}$). For comparison: if we assume $p_R = 0.33$ s.t., $p_E = 0.28$ and $p_M = 0.05$, our attack would require $\epsilon \approx 0.603 \cdot r_e$.

The only comparable transaction exclusion attack is the *Script Puzzle 38.2% attack*, which requires $p_B > 38.2\%$ (in Bitcoin). For comparison, if we assume $p_R = 0.382$, our attacks require a bribe value ϵ close to zero: mining on the attacker chain becomes the highest paying strategy independent of the bribe.

A.6.5 Counter Mechanisms

Unique transaction specification: To deny some transaction from getting into the blockchain, the respective transaction has to be known. We made the simplifying assumption that the transaction hash is known to the attacker and will not change. Although, in practice, this might not hold because there are several counter measures the victim can take. Even if transaction malleability is not possible for any third party, transactions can be recreated by the sender s.t. they are semantically equivalent but their transaction hash differs. Ethereum actively supports this as *replace-by-fee*, when a new transaction from the same account with a higher gas value is available, miners will prefer it. The new transaction is not even required to be semantically equivalent to the original one.

Therefore, the victim can evade the attack if the attack contract relies on transaction hashes. A possible but less generic way around this is to evaluate contract states instead of transaction hashes to determine if the effects of some unwanted transaction have made it into the blockchain. Although this seems like a promising approach, the feasibility of this solution highly depends on the individual implementation of the attack. As outlined in Section A.5.2, the specification of the entire list of transactions by the attacker would not have this problem.

Counter-bribing The most effective countermeasure against the attack, is to increase the fee of tx_V s.t. it surpasses the value promised by the attack contract. For transaction exclusion, the required conditions have to be made public. Therefore this attack cannot be considered stealthy in the target cryptocurrency. This motivates that the operation of the attack is orchestrated from another funding cryptocurrency (utilizing out-of-band payments) and thus hidden from clients that only operate and monitor the target cryptocurrency. Such an attack is described in the Section 4.5

Counter-attack The exact same incentive attack can be used to keep an inclusion proof for the unwanted transaction out of the blockchain. We now show that this is not necessarily a cost efficient counter-attack by introducing an additional cost gap. To introduce this cost gap between the attack and its counter-attack, the stabilization period between e_N and e_T can be increased s.t. it is larger than the period between e_1 and e_N . Thereby, the counter-attack gets more expensive than the original attack. This leverages

the fact that the victim has to get his transaction into the blockchain before e_N , whereas the attacker can choose a longer stabilization period.

Proof a negative If the successful execution of the attack relies on a proof that the unwanted transaction tx_V was included to correctly payout rewards and detect unwanted inclusion, rational miners would be disincentivized to include this proof and keep collecting the rewards for complacent blocks instead.

Therefore, an approach that poses more convincing evidence of transaction absence is desirable. An in-band method that relies on a proof that the transaction tx_V was indeed *not* included in the chain in the respective interval would be ideal. Thereby, the attacker can be sure that the payment only happens if the requested condition is fulfilled. In practice, such proofs are less efficient in current cryptocurrencies such as Ethereum. A possible way around this is to provide a *block template* for every block, which miners must use to be later able to collect the associated additional reward ϵ . Thereby, it can be ensured by the attacker that only wanted transactions are included as well as their order. The block template can be provided in a transaction to an attack contract which encompasses all transaction hashes in their respective order, which should be included in the next block, excluding his own hash.

Another alternative would be to use out-of-band techniques and launch the attack from a different smart contract capable funding cryptocurrency whose miners are not affected by the attack. Moreover, if the set of miners is distinct, the incentives of the miners to not include an inclusion proof of tx_V is less of an issue. We describe an out-of-band attack that uses the technique of *block templates* and also allows for arbitrary ordering in Section 4.5.

A.7 Comparison Tables for “How much is the fork”

In this section, we want to list and compare concrete values of different models and configurations described in Chapter 5. To aid reproducibility and allow readers to spot small changes in values which are not observable in figures, we provide tables for some parameterizations. If the required bribe ϵ in a cell is 0 the cell is colored blue. The code that generates all tables and figures can be found on GitHub⁸.

⁸https://github.com/kernoelpanic/howmuchisthefork_artefacts

	$p_m = 0.05$	$p_m = 0.1$	$p_m = 0.2$	$p_m = 0.3$	$p_m = 0.33$	$p_m = 0.382$	$p_m = 0.4$
$p_{\mathcal{E}} = 0.00$	$p_{\mathcal{A}} = 0.950$ $\rho = 0.050$ $\epsilon = 17.054$ $\rho' = 0.050$ Pr = 0.003	$p_{\mathcal{A}} = 0.900$ $\rho = 0.100$ $\epsilon = 7.125$ $\rho' = 0.100$ Pr = 0.012	$p_{\mathcal{A}} = 0.800$ $\rho = 0.200$ $\epsilon = 2.321$ $\rho' = 0.200$ Pr = 0.060	$p_{\mathcal{A}} = 0.700$ $\rho = 0.300$ $\epsilon = 0.883$ $\rho' = 0.300$ Pr = 0.159	$p_{\mathcal{A}} = 0.670$ $\rho = 0.330$ $\epsilon = 0.649$ $\rho' = 0.330$ Pr = 0.200	$p_{\mathcal{A}} = 0.618$ $\rho = 0.382$ $\epsilon = 0.353$ $\rho' = 0.382$ Pr = 0.282	$p_{\mathcal{A}} = 0.600$ $\rho = 0.400$ $\epsilon = 0.275$ $\rho' = 0.400$ Pr = 0.314
$p_{\mathcal{E}} = 0.05$	$p_{\mathcal{A}} = 0.900$ $\rho = 0.052$ $\epsilon = 7.529$ $\rho' = 0.052$ Pr = 0.012	$p_{\mathcal{A}} = 0.850$ $\rho = 0.105$ $\epsilon = 4.126$ $\rho' = 0.105$ Pr = 0.031	$p_{\mathcal{A}} = 0.750$ $\rho = 0.210$ $\epsilon = 1.556$ $\rho' = 0.210$ Pr = 0.103	$p_{\mathcal{A}} = 0.650$ $\rho = 0.315$ $\epsilon = 0.596$ $\rho' = 0.315$ Pr = 0.230	$p_{\mathcal{A}} = 0.620$ $\rho = 0.346$ $\epsilon = 0.430$ $\rho' = 0.346$ Pr = 0.279	$p_{\mathcal{A}} = 0.568$ $\rho = 0.401$ $\epsilon = 0.216$ $\rho' = 0.401$ Pr = 0.373	$p_{\mathcal{A}} = 0.550$ $\rho = 0.420$ $\epsilon = 0.159$ $\rho' = 0.420$ Pr = 0.408
$p_{\mathcal{E}} = 0.10$	$p_{\mathcal{A}} = 0.850$ $\rho = 0.055$ $\epsilon = 4.359$ $\rho' = 0.055$ Pr = 0.031	$p_{\mathcal{A}} = 0.800$ $\rho = 0.110$ $\epsilon = 2.645$ $\rho' = 0.110$ Pr = 0.060	$p_{\mathcal{A}} = 0.700$ $\rho = 0.219$ $\epsilon = 1.067$ $\rho' = 0.219$ Pr = 0.159	$p_{\mathcal{A}} = 0.600$ $\rho = 0.329$ $\epsilon = 0.399$ $\rho' = 0.329$ Pr = 0.314	$p_{\mathcal{A}} = 0.570$ $\rho = 0.362$ $\epsilon = 0.278$ $\rho' = 0.362$ Pr = 0.369	$p_{\mathcal{A}} = 0.518$ $\rho = 0.419$ $\epsilon = 0.122$ $\rho' = 0.419$ Pr = 0.471	$p_{\mathcal{A}} = 0.500$ $\rho = 0.439$ $\epsilon = 0.081$ $\rho' = 0.439$ Pr = 0.508
$p_{\mathcal{E}} = 0.20$	$p_{\mathcal{A}} = 0.750$ $\rho = 0.059$ $\epsilon = 1.860$ $\rho' = 0.059$ Pr = 0.103	$p_{\mathcal{A}} = 0.700$ $\rho = 0.117$ $\epsilon = 1.212$ $\rho' = 0.117$ Pr = 0.159	$p_{\mathcal{A}} = 0.600$ $\rho = 0.235$ $\epsilon = 0.497$ $\rho' = 0.235$ Pr = 0.314	$p_{\mathcal{A}} = 0.500$ $\rho = 0.352$ $\epsilon = 0.157$ $\rho' = 0.352$ Pr = 0.508	$p_{\mathcal{A}} = 0.470$ $\rho = 0.388$ $\epsilon = 0.095$ $\rho' = 0.388$ Pr = 0.569	$p_{\mathcal{A}} = 0.418$ $\rho = 0.449$ $\epsilon = 0.017$ $\rho' = 0.449$ Pr = 0.672	$p_{\mathcal{A}} = 0.400$ $\rho = 0.470$ $\epsilon = 0.000$ $\rho' = 0.471$ Pr = 0.706
$p_{\mathcal{E}} = 0.30$	$p_{\mathcal{A}} = 0.650$ $\rho = 0.060$ $\epsilon = 0.827$ $\rho' = 0.060$ Pr = 0.230	$p_{\mathcal{A}} = 0.600$ $\rho = 0.120$ $\epsilon = 0.531$ $\rho' = 0.120$ Pr = 0.314	$p_{\mathcal{A}} = 0.500$ $\rho = 0.240$ $\epsilon = 0.183$ $\rho' = 0.240$ Pr = 0.508	$p_{\mathcal{A}} = 0.400$ $\rho = 0.360$ $\epsilon = 0.020$ $\rho' = 0.360$ Pr = 0.706	$p_{\mathcal{A}} = 0.370$ $\rho = 0.396$ $\epsilon = 0.000$ $\rho' = 0.398$ Pr = 0.760	$p_{\mathcal{A}} = 0.318$ $\rho = 0.459$ $\epsilon = 0.000$ $\rho' = 0.472$ Pr = 0.843	$p_{\mathcal{A}} = 0.300$ $\rho = 0.480$ $\epsilon = 0.000$ $\rho' = 0.496$ Pr = 0.867
$p_{\mathcal{E}} = 0.33$	$p_{\mathcal{A}} = 0.620$ $\rho = 0.060$ $\epsilon = 0.626$ $\rho' = 0.060$ Pr = 0.279	$p_{\mathcal{A}} = 0.570$ $\rho = 0.119$ $\epsilon = 0.390$ $\rho' = 0.119$ Pr = 0.369	$p_{\mathcal{A}} = 0.470$ $\rho = 0.239$ $\epsilon = 0.112$ $\rho' = 0.239$ Pr = 0.569	$p_{\mathcal{A}} = 0.370$ $\rho = 0.358$ $\epsilon = 0.000$ $\rho' = 0.362$ Pr = 0.760	$p_{\mathcal{A}} = 0.340$ $\rho = 0.394$ $\epsilon = 0.000$ $\rho' = 0.405$ Pr = 0.810	$p_{\mathcal{A}} = 0.288$ $\rho = 0.456$ $\epsilon = 0.000$ $\rho' = 0.474$ Pr = 0.883	$p_{\mathcal{A}} = 0.270$ $\rho = 0.478$ $\epsilon = 0.000$ $\rho' = 0.495$ Pr = 0.904
$p_{\mathcal{E}} = 0.38$	$p_{\mathcal{A}} = 0.568$ $\rho = 0.058$ $\epsilon = 0.345$ $\rho' = 0.058$ Pr = 0.373	$p_{\mathcal{A}} = 0.518$ $\rho = 0.116$ $\epsilon = 0.187$ $\rho' = 0.116$ Pr = 0.471	$p_{\mathcal{A}} = 0.418$ $\rho = 0.232$ $\epsilon = 0.006$ $\rho' = 0.232$ Pr = 0.672	$p_{\mathcal{A}} = 0.318$ $\rho = 0.348$ $\epsilon = 0.000$ $\rho' = 0.371$ Pr = 0.843	$p_{\mathcal{A}} = 0.288$ $\rho = 0.383$ $\epsilon = 0.000$ $\rho' = 0.409$ Pr = 0.883	$p_{\mathcal{A}} = 0.236$ $\rho = 0.444$ $\epsilon = 0.000$ $\rho' = 0.468$ Pr = 0.937	$p_{\mathcal{A}} = 0.218$ $\rho = 0.464$ $\epsilon = 0.000$ $\rho' = 0.487$ Pr = 0.951
$p_{\mathcal{E}} = 0.40$	$p_{\mathcal{A}} = 0.550$ $\rho = 0.057$ $\epsilon = 0.262$ $\rho' = 0.057$ Pr = 0.408	$p_{\mathcal{A}} = 0.500$ $\rho = 0.114$ $\epsilon = 0.126$ $\rho' = 0.114$ Pr = 0.508	$p_{\mathcal{A}} = 0.400$ $\rho = 0.229$ $\epsilon = 0.000$ $\rho' = 0.235$ Pr = 0.706	$p_{\mathcal{A}} = 0.300$ $\rho = 0.343$ $\epsilon = 0.000$ $\rho' = 0.372$ Pr = 0.867	$p_{\mathcal{A}} = 0.270$ $\rho = 0.377$ $\epsilon = 0.000$ $\rho' = 0.408$ Pr = 0.904	$p_{\mathcal{A}} = 0.218$ $\rho = 0.437$ $\epsilon = 0.000$ $\rho' = 0.465$ Pr = 0.951	$p_{\mathcal{A}} = 0.200$ $\rho = 0.457$ $\epsilon = 0.000$ $\rho' = 0.482$ Pr = 0.963

Table A.1: Comparison of minimum bribe required per block ϵ for $\rho_{fork\ bl.} > \rho_{main\ bl.}$ with probabilities calculated using our Markov chain. The axis iterate the hashrate of an individual miner p_m , and other attackers $p_{\mathcal{E}}$. The table also shows the expected reward of miner m , if p_m would be directed towards the attack chain $\rho' = \rho_{fork\ bl.}$, as well as the expected reward $\rho = \rho_{main\ bl.}$, if p_m would be directed towards the main chain. All attacks start with a disadvantage of $z = 1$, a duration of $N = 8$ and the following configuration of the Markov chain $\vec{k} = 1, \overleftarrow{k} = 10, \eta_{attack} = 0, \eta_{main} = 0, p_{\mathcal{V}} = 0.000000$.

A.7. Comparison Tables for “How much is the fork”

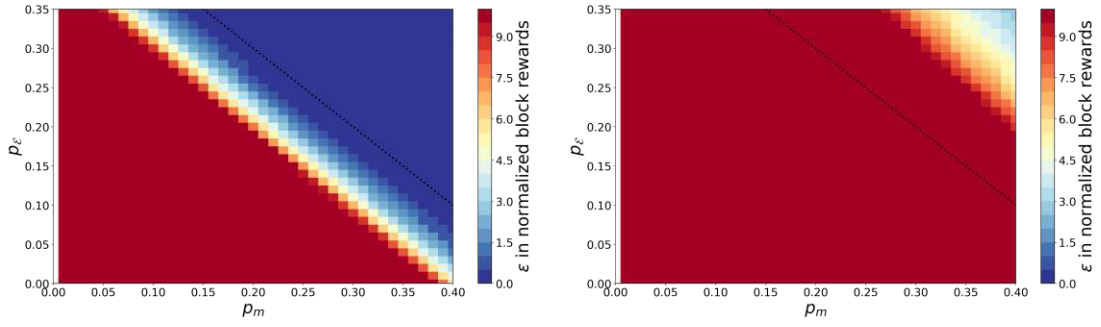
	$p_m = 0.05$	$p_m = 0.1$	$p_m = 0.2$	$p_m = 0.3$	$p_m = 0.33$	$p_m = 0.382$	$p_m = 0.4$
$p_{\mathcal{E}} = 0.00$	$p_A = 0.850$ $\rho = 1.050$ $\epsilon = 20.157$ $\rho' = 1.050$ Pr = 0.002	$p_A = 0.800$ $\rho = 1.100$ $\epsilon = 9.004$ $\rho' = 1.100$ Pr = 0.011	$p_A = 0.700$ $\rho = 1.200$ $\epsilon = 3.617$ $\rho' = 1.200$ Pr = 0.055	$p_A = 0.600$ $\rho = 1.300$ $\epsilon = 2.015$ $\rho' = 1.300$ Pr = 0.149	$p_A = 0.570$ $\rho = 1.330$ $\epsilon = 1.755$ $\rho' = 1.330$ Pr = 0.188	$p_A = 0.518$ $\rho = 1.382$ $\epsilon = 1.429$ $\rho' = 1.382$ Pr = 0.267	$p_A = 0.500$ $\rho = 1.400$ $\epsilon = 1.342$ $\rho' = 1.400$ Pr = 0.298
$p_{\mathcal{E}} = 0.05$	$p_A = 0.800$ $\rho = 1.050$ $\epsilon = 10.039$ $\rho' = 1.050$ Pr = 0.011	$p_A = 0.750$ $\rho = 1.103$ $\epsilon = 6.021$ $\rho' = 1.103$ Pr = 0.028	$p_A = 0.650$ $\rho = 1.208$ $\epsilon = 2.988$ $\rho' = 1.208$ Pr = 0.095	$p_A = 0.550$ $\rho = 1.314$ $\epsilon = 1.856$ $\rho' = 1.314$ Pr = 0.217	$p_A = 0.520$ $\rho = 1.346$ $\epsilon = 1.659$ $\rho' = 1.346$ Pr = 0.264	$p_A = 0.468$ $\rho = 1.401$ $\epsilon = 1.404$ $\rho' = 1.401$ Pr = 0.356	$p_A = 0.450$ $\rho = 1.420$ $\epsilon = 1.336$ $\rho' = 1.420$ Pr = 0.390
$p_{\mathcal{E}} = 0.10$	$p_A = 0.750$ $\rho = 1.045$ $\epsilon = 6.781$ $\rho' = 1.045$ Pr = 0.028	$p_A = 0.700$ $\rho = 1.100$ $\epsilon = 4.630$ $\rho' = 1.100$ Pr = 0.055	$p_A = 0.600$ $\rho = 1.212$ $\epsilon = 2.639$ $\rho' = 1.212$ Pr = 0.149	$p_A = 0.500$ $\rho = 1.325$ $\epsilon = 1.786$ $\rho' = 1.325$ Pr = 0.298	$p_A = 0.470$ $\rho = 1.358$ $\epsilon = 1.630$ $\rho' = 1.358$ Pr = 0.352	$p_A = 0.418$ $\rho = 1.417$ $\epsilon = 1.424$ $\rho' = 1.417$ Pr = 0.453	$p_A = 0.400$ $\rho = 1.437$ $\epsilon = 1.369$ $\rho' = 1.437$ Pr = 0.489
$p_{\mathcal{E}} = 0.20$	$p_A = 0.650$ $\rho = 1.007$ $\epsilon = 4.355$ $\rho' = 1.007$ Pr = 0.095	$p_A = 0.600$ $\rho = 1.069$ $\epsilon = 3.398$ $\rho' = 1.069$ Pr = 0.149	$p_A = 0.500$ $\rho = 1.196$ $\epsilon = 2.319$ $\rho' = 1.196$ Pr = 0.298	$p_A = 0.400$ $\rho = 1.326$ $\epsilon = 1.777$ $\rho' = 1.326$ Pr = 0.489	$p_A = 0.370$ $\rho = 1.365$ $\epsilon = 1.673$ $\rho' = 1.365$ Pr = 0.550	$p_A = 0.318$ $\rho = 1.433$ $\epsilon = 1.534$ $\rho' = 1.433$ Pr = 0.653	$p_A = 0.300$ $\rho = 1.457$ $\epsilon = 1.496$ $\rho' = 1.457$ Pr = 0.688
$p_{\mathcal{E}} = 0.30$	$p_A = 0.550$ $\rho = 0.918$ $\epsilon = 3.365$ $\rho' = 0.918$ Pr = 0.217	$p_A = 0.500$ $\rho = 0.987$ $\epsilon = 2.827$ $\rho' = 0.987$ Pr = 0.298	$p_A = 0.400$ $\rho = 1.129$ $\epsilon = 2.160$ $\rho' = 1.129$ Pr = 0.489	$p_A = 0.300$ $\rho = 1.277$ $\epsilon = 1.805$ $\rho' = 1.277$ Pr = 0.688	$p_A = 0.270$ $\rho = 1.322$ $\epsilon = 1.737$ $\rho' = 1.322$ Pr = 0.743	$p_A = 0.218$ $\rho = 1.402$ $\epsilon = 1.652$ $\rho' = 1.402$ Pr = 0.828	$p_A = 0.200$ $\rho = 1.430$ $\epsilon = 1.631$ $\rho' = 1.430$ Pr = 0.854
$p_{\mathcal{E}} = 0.33$	$p_A = 0.520$ $\rho = 0.880$ $\epsilon = 3.153$ $\rho' = 0.880$ Pr = 0.264	$p_A = 0.470$ $\rho = 0.950$ $\epsilon = 2.692$ $\rho' = 0.950$ Pr = 0.352	$p_A = 0.370$ $\rho = 1.096$ $\epsilon = 2.112$ $\rho' = 1.096$ Pr = 0.550	$p_A = 0.270$ $\rho = 1.249$ $\epsilon = 1.803$ $\rho' = 1.249$ Pr = 0.743	$p_A = 0.240$ $\rho = 1.296$ $\epsilon = 1.745$ $\rho' = 1.296$ Pr = 0.794	$p_A = 0.188$ $\rho = 1.379$ $\epsilon = 1.676$ $\rho' = 1.379$ Pr = 0.870	$p_A = 0.170$ $\rho = 1.408$ $\epsilon = 1.660$ $\rho' = 1.408$ Pr = 0.892
$p_{\mathcal{E}} = 0.38$	$p_A = 0.468$ $\rho = 0.801$ $\epsilon = 2.818$ $\rho' = 0.801$ Pr = 0.356	$p_A = 0.418$ $\rho = 0.873$ $\epsilon = 2.464$ $\rho' = 0.873$ Pr = 0.453	$p_A = 0.318$ $\rho = 1.023$ $\epsilon = 2.011$ $\rho' = 1.023$ Pr = 0.653	$p_A = 0.218$ $\rho = 1.183$ $\epsilon = 1.775$ $\rho' = 1.183$ Pr = 0.828	$p_A = 0.188$ $\rho = 1.233$ $\epsilon = 1.735$ $\rho' = 1.233$ Pr = 0.870	$p_A = 0.136$ $\rho = 1.321$ $\epsilon = 1.693$ $\rho' = 1.321$ Pr = 0.927	$p_A = 0.118$ $\rho = 1.352$ $\epsilon = 1.685$ $\rho' = 1.352$ Pr = 0.943
$p_{\mathcal{E}} = 0.40$	$p_A = 0.450$ $\rho = 0.771$ $\epsilon = 2.706$ $\rho' = 0.771$ Pr = 0.390	$p_A = 0.400$ $\rho = 0.842$ $\epsilon = 2.383$ $\rho' = 0.842$ Pr = 0.489	$p_A = 0.300$ $\rho = 0.993$ $\epsilon = 1.968$ $\rho' = 0.993$ Pr = 0.688	$p_A = 0.200$ $\rho = 1.155$ $\epsilon = 1.757$ $\rho' = 1.155$ Pr = 0.854	$p_A = 0.170$ $\rho = 1.206$ $\epsilon = 1.722$ $\rho' = 1.206$ Pr = 0.892	$p_A = 0.118$ $\rho = 1.296$ $\epsilon = 1.689$ $\rho' = 1.296$ Pr = 0.943	$p_A = 0.100$ $\rho = 1.328$ $\epsilon = 1.685$ $\rho' = 1.328$ Pr = 0.956

Table A.2: Comparison of minimum bribe required per block ϵ for $\rho_{fork\ bl.} > \rho_{main\ bl.}$ with probabilities calculated using our Markov chain. The axis iterate the hashrate of an individual miner p_m , and other attackers $p_{\mathcal{E}}$. The table also shows the expected reward of miner m , if p_m would be directed towards the attack chain $\rho' = \rho_{fork\ bl.}$, as well as the expected reward $\rho = \rho_{main\ bl.}$, if p_m would be directed towards the main chain. All attacks start with a disadvantage of $z = 1$, a duration of $N = 8$ and the following configuration of the Markov chain $\vec{k} = 3, \overleftarrow{k} = 10, \eta_{attack} = 0, \eta_{main} = 1, p_{\mathcal{V}} = 0.100000$.

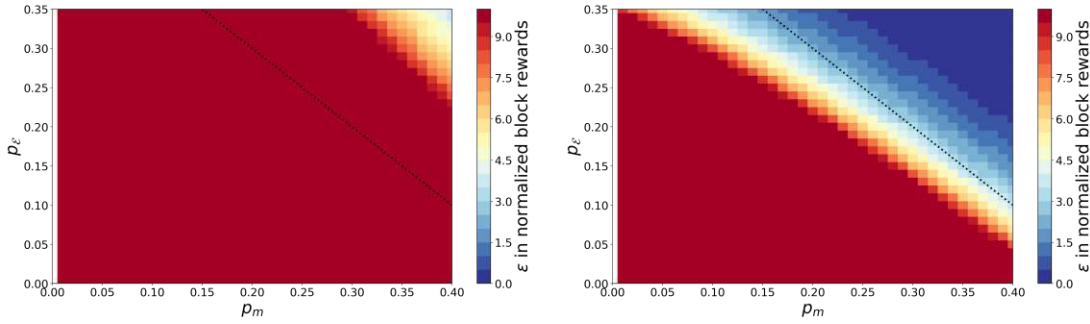
	$p_m = 0.05$	$p_m = 0.1$	$p_m = 0.2$	$p_m = 0.3$	$p_m = 0.33$	$p_m = 0.382$	$p_m = 0.4$
$p_{\mathcal{E}} = 0.00$	$p_{\mathcal{A}} = 0.850$ $\rho = 0.050$ $\epsilon = 0.000$ $\rho' = 1.000$ Pr = 0.000	$p_{\mathcal{A}} = 0.800$ $\rho = 0.100$ $\epsilon = 0.000$ $\rho' = 1.000$ Pr = 0.000	$p_{\mathcal{A}} = 0.700$ $\rho = 0.200$ $\epsilon = 0.000$ $\rho' = 1.000$ Pr = 0.000	$p_{\mathcal{A}} = 0.600$ $\rho = 0.300$ $\epsilon = 0.000$ $\rho' = 1.000$ Pr = 0.000	$p_{\mathcal{A}} = 0.570$ $\rho = 0.330$ $\epsilon = 0.000$ $\rho' = 1.000$ Pr = 0.000	$p_{\mathcal{A}} = 0.518$ $\rho = 0.382$ $\epsilon = 0.000$ $\rho' = 1.000$ Pr = 0.001	$p_{\mathcal{A}} = 0.500$ $\rho = 0.400$ $\epsilon = 0.000$ $\rho' = 1.000$ Pr = 0.001
$p_{\mathcal{E}} = 0.05$	$p_{\mathcal{A}} = 0.800$ $\rho = 0.053$ $\epsilon = 0.000$ $\rho' = 0.500$ Pr = 0.000	$p_{\mathcal{A}} = 0.750$ $\rho = 0.105$ $\epsilon = 0.000$ $\rho' = 0.667$ Pr = 0.000	$p_{\mathcal{A}} = 0.650$ $\rho = 0.211$ $\epsilon = 0.000$ $\rho' = 0.800$ Pr = 0.000	$p_{\mathcal{A}} = 0.550$ $\rho = 0.316$ $\epsilon = 0.000$ $\rho' = 0.857$ Pr = 0.001	$p_{\mathcal{A}} = 0.520$ $\rho = 0.347$ $\epsilon = 0.000$ $\rho' = 0.868$ Pr = 0.001	$p_{\mathcal{A}} = 0.468$ $\rho = 0.402$ $\epsilon = 0.000$ $\rho' = 0.884$ Pr = 0.003	$p_{\mathcal{A}} = 0.450$ $\rho = 0.421$ $\epsilon = 0.000$ $\rho' = 0.889$ Pr = 0.003
$p_{\mathcal{E}} = 0.10$	$p_{\mathcal{A}} = 0.750$ $\rho = 0.056$ $\epsilon = 0.000$ $\rho' = 0.333$ Pr = 0.000	$p_{\mathcal{A}} = 0.700$ $\rho = 0.111$ $\epsilon = 0.000$ $\rho' = 0.500$ Pr = 0.000	$p_{\mathcal{A}} = 0.600$ $\rho = 0.222$ $\epsilon = 0.000$ $\rho' = 0.667$ Pr = 0.000	$p_{\mathcal{A}} = 0.500$ $\rho = 0.333$ $\epsilon = 0.000$ $\rho' = 0.750$ Pr = 0.001	$p_{\mathcal{A}} = 0.470$ $\rho = 0.367$ $\epsilon = 0.000$ $\rho' = 0.767$ Pr = 0.002	$p_{\mathcal{A}} = 0.418$ $\rho = 0.424$ $\epsilon = 0.000$ $\rho' = 0.793$ Pr = 0.005	$p_{\mathcal{A}} = 0.400$ $\rho = 0.444$ $\epsilon = 0.000$ $\rho' = 0.800$ Pr = 0.007
$p_{\mathcal{E}} = 0.20$	$p_{\mathcal{A}} = 0.650$ $\rho = 0.062$ $\epsilon = 0.000$ $\rho' = 0.200$ Pr = 0.000	$p_{\mathcal{A}} = 0.600$ $\rho = 0.125$ $\epsilon = 0.000$ $\rho' = 0.333$ Pr = 0.000	$p_{\mathcal{A}} = 0.500$ $\rho = 0.250$ $\epsilon = 0.000$ $\rho' = 0.500$ Pr = 0.001	$p_{\mathcal{A}} = 0.400$ $\rho = 0.375$ $\epsilon = 0.000$ $\rho' = 0.600$ Pr = 0.007	$p_{\mathcal{A}} = 0.370$ $\rho = 0.412$ $\epsilon = 0.000$ $\rho' = 0.623$ Pr = 0.011	$p_{\mathcal{A}} = 0.318$ $\rho = 0.477$ $\epsilon = 0.000$ $\rho' = 0.656$ Pr = 0.020	$p_{\mathcal{A}} = 0.300$ $\rho = 0.500$ $\epsilon = 0.000$ $\rho' = 0.667$ Pr = 0.025
$p_{\mathcal{E}} = 0.30$	$p_{\mathcal{A}} = 0.550$ $\rho = 0.071$ $\epsilon = 0.000$ $\rho' = 0.143$ Pr = 0.001	$p_{\mathcal{A}} = 0.500$ $\rho = 0.143$ $\epsilon = 0.000$ $\rho' = 0.250$ Pr = 0.001	$p_{\mathcal{A}} = 0.400$ $\rho = 0.286$ $\epsilon = 0.000$ $\rho' = 0.400$ Pr = 0.007	$p_{\mathcal{A}} = 0.300$ $\rho = 0.429$ $\epsilon = 0.000$ $\rho' = 0.500$ Pr = 0.025	$p_{\mathcal{A}} = 0.270$ $\rho = 0.471$ $\epsilon = 0.000$ $\rho' = 0.524$ Pr = 0.035	$p_{\mathcal{A}} = 0.218$ $\rho = 0.546$ $\epsilon = 0.000$ $\rho' = 0.560$ Pr = 0.062	$p_{\mathcal{A}} = 0.200$ $\rho = 0.571$ $\epsilon = 0.000$ $\rho' = 0.571$ Pr = 0.074
$p_{\mathcal{E}} = 0.33$	$p_{\mathcal{A}} = 0.520$ $\rho = 0.075$ $\epsilon = 0.000$ $\rho' = 0.132$ Pr = 0.001	$p_{\mathcal{A}} = 0.470$ $\rho = 0.149$ $\epsilon = 0.000$ $\rho' = 0.233$ Pr = 0.002	$p_{\mathcal{A}} = 0.370$ $\rho = 0.298$ $\epsilon = 0.000$ $\rho' = 0.377$ Pr = 0.011	$p_{\mathcal{A}} = 0.270$ $\rho = 0.448$ $\epsilon = 0.000$ $\rho' = 0.476$ Pr = 0.035	$p_{\mathcal{A}} = 0.240$ $\rho = 0.492$ $\epsilon = 0.000$ $\rho' = 0.500$ Pr = 0.049	$p_{\mathcal{A}} = 0.188$ $\rho = 0.570$ $\epsilon = 0.748$ $\rho' = 0.570$ Pr = 0.083	$p_{\mathcal{A}} = 0.170$ $\rho = 0.597$ $\epsilon = 0.898$ $\rho' = 0.597$ Pr = 0.099
$p_{\mathcal{E}} = 0.38$	$p_{\mathcal{A}} = 0.468$ $\rho = 0.081$ $\epsilon = 0.000$ $\rho' = 0.116$ Pr = 0.003	$p_{\mathcal{A}} = 0.418$ $\rho = 0.162$ $\epsilon = 0.000$ $\rho' = 0.207$ Pr = 0.005	$p_{\mathcal{A}} = 0.318$ $\rho = 0.323$ $\epsilon = 0.000$ $\rho' = 0.344$ Pr = 0.020	$p_{\mathcal{A}} = 0.218$ $\rho = 0.485$ $\epsilon = 1.664$ $\rho' = 0.485$ Pr = 0.062	$p_{\mathcal{A}} = 0.188$ $\rho = 0.534$ $\epsilon = 1.813$ $\rho' = 0.534$ Pr = 0.083	$p_{\mathcal{A}} = 0.136$ $\rho = 0.618$ $\epsilon = 1.722$ $\rho' = 0.618$ Pr = 0.137	$p_{\mathcal{A}} = 0.118$ $\rho = 0.647$ $\epsilon = 1.644$ $\rho' = 0.647$ Pr = 0.161
$p_{\mathcal{E}} = 0.40$	$p_{\mathcal{A}} = 0.450$ $\rho = 0.083$ $\epsilon = 0.000$ $\rho' = 0.111$ Pr = 0.003	$p_{\mathcal{A}} = 0.400$ $\rho = 0.166$ $\epsilon = 0.000$ $\rho' = 0.200$ Pr = 0.007	$p_{\mathcal{A}} = 0.300$ $\rho = 0.333$ $\epsilon = 0.000$ $\rho' = 0.333$ Pr = 0.025	$p_{\mathcal{A}} = 0.200$ $\rho = 0.500$ $\epsilon = 2.233$ $\rho' = 0.500$ Pr = 0.074	$p_{\mathcal{A}} = 0.170$ $\rho = 0.549$ $\epsilon = 2.168$ $\rho' = 0.549$ Pr = 0.099	$p_{\mathcal{A}} = 0.118$ $\rho = 0.636$ $\epsilon = 1.878$ $\rho' = 0.636$ Pr = 0.161	$p_{\mathcal{A}} = 0.100$ $\rho = 0.666$ $\epsilon = 1.760$ $\rho' = 0.666$ Pr = 0.189

Table A.3: Comparison of minimum bribe required per block ϵ for $\rho_{fork\ comp.} > \rho_{main\ comp.}$ with effort-related compensation and probabilities calculated using our Markov chain. The axis iterate the hashrate of an individual miner p_m , and other attackers $p_{\mathcal{E}}$. The table also shows the expected reward of miner m , if p_m would be directed towards the attack chain $\rho' = \rho_{fork\ comp.}$, as well as the expected reward $\rho = \rho_{main\ comp.}$, if p_m would be directed towards the main chain. All attacks start with a disadvantage of $z = 6$ and a duration of $N = 8$ and the following configuration of the Markov chain $\vec{k} = 3, \overleftarrow{k} = 10, \eta_{attack} = 0, \eta_{main} = 0, p_{\mathcal{V}} = 0.100000$.

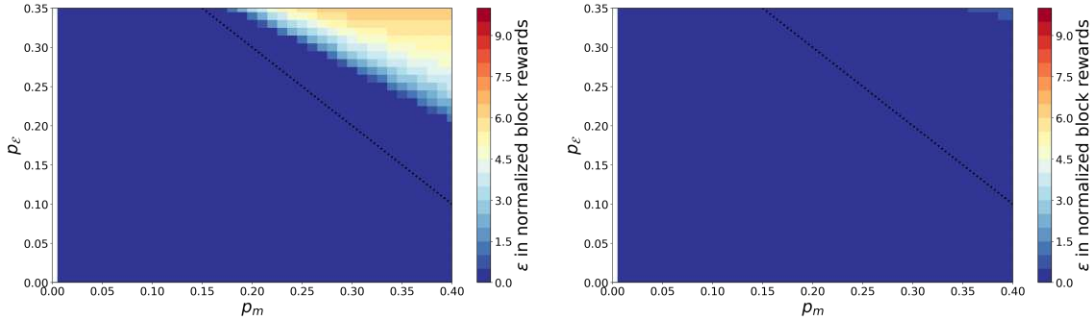
A.7. Comparison Tables for “How much is the fork”



(a) Minimum ϵ in classic model with $z = 6, N = \infty$ (b) Minimum ϵ in Markov model with $p_V = 0, z = 6, \vec{k} = 1, \overleftarrow{k} = 9, N = 10$



(c) Minimum ϵ in Markov model with $p_V = 0.25, z = 6, \vec{k} = 3, \overleftarrow{k} = 9, N = 10$ (d) Minimum ϵ in Markov model with contributed blocks $p_V = 0.25, z = 6, \vec{k} = 3, \overleftarrow{k} = 9, N = 10, \eta_{attack} = 3$



(e) Minimum ϵ in Markov model with effort-related compensation $p_V = 0.25, z = 6, \vec{k} = 3, \overleftarrow{k} = 9, N = 10, \eta_{main} = 3$ (f) Minimum ϵ in Markov model with effort-related compensation $p_V = 0.25, z = 6, \vec{k} = 3, \overleftarrow{k} = 9, N = 10, \eta_{main} = 0$

Figure A.3: Minimum bribe value ϵ in normalized block rewards for different models and configurations.

A.8 Comparison of Other Definitions of Extractable Value from Related Work

In this section, we outline concurrent work on defining different forms of extractable value.

A.8.1 Clockwork Finance

In concurrent work [BDKJ21] a formal definition of MEV was provided within the *clockwork finance framework* (CFF), which uses “*formal verification techniques to reason about the profit extractable from the system by a participant*” [BDKJ21]. The goal of CFF is it to prove bounds on the economic security of certain contracts without manually coding adversarial strategies, which is defined as the “*maximum amount adversaries can extract from them*” [BDKJ21]. Therefore, they use the K Framework [Ros17], an executable semantic framework in which they modeled certain interesting DeFi contracts and reasoned about the maximum extractable value they offer if deployed together on the same system. They validated the modeled contracts empirically in a very elegant way, by utilizing the capabilities of K to use concrete inputs to programs to be evaluated. Therefore, they were able to replay previously collected real-world data of interactions with the respective contracts to the modeled contracts and compare the final states. Furthermore, they provide scripts to download, process, and validate this data for each analyzed protocol.

Clearly, this work is of practical importance for quickly discovering and analyzing attacks on deployed and new DeFi contracts and protocols, which result in a situation where attackers can extract high amounts of value if certain contracts are deployed/used in combination. This property is referred to as the *composability* of contracts in [BDKJ21]. However, the chosen approach and model also face certain limitations and challenges, which we will discuss in more detail after summarizing the original paper.

In the paper [BDKJ21] *extractable value* (EV) is defined as the “*maximum value, expressed in terms of the primary token, that can be extracted by a given player from a valid sequence of blocks that extends the current chain.*”. More formally, the extractable value for a player P , with accounts A_P , starting from a state s and given a valid block sequence \mathcal{B} of length k is defined by the maximum achievable balance in the primary token on all his accounts after k blocks. A summary of relevant variables and symbols according to the model from [BDKJ21] is given in Table A.4.

$$\text{EV}(P, \mathcal{B}, s) = \max_{(B_1, \dots, B_k) \in \mathcal{B}} \left(\sum_{a \in A_P} \text{balance}_k(a)[0] - \text{balance}_0(a)[0] \right) \quad (\text{A.3})$$

From this definition of EV, they define the *miner extractable value* (MEV or k-MEV) if P is a miner as the EV from *all* valid blocks that can be created by P in state s :

$$\text{MEV}(P, s) = \text{EV}(P, \text{validBlocks}_1(P, s), s) \quad (\text{A.4})$$

$$k\text{-MEV}(P, s) = \text{EV}(P, \text{validBlocks}_k(P, s), s) \quad (\text{A.5})$$

In other words, the miner extractable value of player P , is the maximum value that can be extracted from all valid blocks that can be created by P in state s . Note that these definitions assume that P is going to mine the next or the next k blocks, but so far, the definition does not account for the probability or the difficulty of this to happen. To capture the probability to mine the exactly k consecutive blocks, the concept of *weighted* MEV (WMEV) is defined in the appendix of the respective paper [BDKJ21]:

$$\text{WMEV}(P, s) = \sum_{k=1}^{\infty} p_k \cdot k\text{-MEV}(P, s) \quad (\text{A.6})$$

With the concept of MEV, *DeFi Composability* is defined as follows in the paper [BDKJ21]:

(DeFi Composability): Consider a state s and a player P , A DeFi instrument C' is ε -composable under (P, s) if

$$\text{MEV}(P, s') \leq (1 + \varepsilon)\text{MEV}(P, s) \quad (\text{A.7})$$

Here s' refers to the state of the system if contract C' is added to state s . In other words, this definition states that a certain contract C' is DeFi composable with a system in state s , for a given player P , if this player cannot gain significantly more value out of the system when C' is deployed on s .

Challenges, Open Problems and the Relation to our Model

In this section, we discuss the differences in the definitions of EV and MEV given in the paper [BDKJ21] to the paper [JSSW22a]. Furthermore, we comment on some challenges and open problems that arise from the definitions of EV and MEV introduced in the paper [BDKJ21] and their relation to the paper at hand. The discussion is continued in Section A.8.2, after summarizing a follow-up work [OSS⁺21] in Section A.8.2, which extends upon certain aspects of the model presented in [BDKJ21] and also provides some open problems in this space.

Definitions: The main differences between the chosen definition of EV in the paper [BDKJ21] the paper [JSSW22a], are the explicit definition of the *current state* (s) as a parameter of the function, whereas the state in our definitions (3,4,5) is defined implicitly by the entire history of the blockchain, i.e., all its transactions which represent the individual state transitions.

Variable	Description
P	A player
\mathcal{B}	Set of valid block sequences of length k
k	Length of specified block sequence \mathcal{B}
s	State of the system
B_1, \dots, B_k	Blocks in \mathcal{B}
A_P	Set of accounts belonging to player P
$a \in A_P$	Individual account of player P
$\text{balance}_x(a)[0]$	Function that returns the balance of a given account a at height x measured in the primary token ($[0]$), e.g., ether.
$\text{data}_x(a)$	Function returning the data of a given account a at height x
$\text{validBlocks}_k(P, s)$	Denotes the set of <i>all</i> valid blocks (or block sequences length k) that can be created by player P in state s if it could work as a miner.
p_k	Probability that P mines exactly k consecutive blocks
s'	State of the system, when contract C' is added to s

Table A.4: Overview of necessary notation and variables used in the model presented in [BDKJ21].

The other significant difference lies in the definition of EV as the *maximum* extractable value by a particular participant, which is implicitly assumed to be one self. From the perspective of the paper [BDKJ21] this makes sense as they aim to operate on unconfirmed transactions since no-forks are considered. Although, this makes this definition not directly applicable when talking about the extractable value of already settled blocks, although this could be augmented.

Furthermore, it should be noticed that identifying the *maximum* EV of future blocks requires perfect information regarding the set of available transactions. As this set is dependent on the transactions other players make, it cannot readily be predicted, especially for multiple blocks into the future. This indicates that, although searching and approximating the maximum achievable extractable value makes sense from a *personal* profit-oriented perspective, defining EV and MEV as *the maximum value* that could be extracted is not optimal given all the computational and practical limitations, as well as for reasoning about potential consensus instability issues.

Combinatorial explosion and required information: As already noted in [BDKJ21], the number of possible orderings of transactions grows factorial in the number of transactions in the mempool. Given that blocks in Ethereum approximately contain 200-250

transactions, reasoning over all possible orderings is computationally infeasible. In the paper [BDKJ21] this problem is tackled by only considering transactions that are relevant (i.e., interact) with the contracts under investigation. Therefore the set of transactions that was considered per block is reduced to 7 to 10 transactions. This is a reasonable approach, as current block sizes and intervals in Ethereum with around 15 transactions per second (TPS) lead to a relatively low number of relevant interactions with specific smart contracts (or a set of smart contracts). However, this restriction may not apply to future versions of Ethereum, which aim to scale up the number of transactions per second. Moreover, such restrictions might not be applicable to current high-throughput alternatives such as Solana⁹ or polygon¹⁰, which claim to handle thousands of transactions per second. So identifying the *maximum* value that can be extracted from a specific set of transactions is a computational problem that is difficult to solve in practice if systems and TPS grow larger.

Potential dependencies between certain transactions might, on-first sight, reduce the overall search space. However, complicated transaction dependencies might still increase the overall computational costs for identifying a valid profitable ordering. As not all reorderings are valid by the protocol rules, the validity of each ordering has to be checked as well, which further increases the overall computational costs as it potentially requires the evaluation of the entire ordering. Although this can be augmented, the validity of profitable orderings identified with CFF is not automatically checked at the moment.

Personal MEV, and MEV of other players: In the paper [BDKJ21], MEV is mostly viewed from the a single player’s perspective, which is implicitly assumed to be one self. This is reasonable and makes sense if someone wants to discover lucrative arbitrage/value extraction opportunities, which has been demonstrated to be possible with CFF. However, this analysis method cannot prove the absence of MEV exploitation vectors for any other player in the system, as their capabilities, as well as their accounts, might not be known to a third party. Therefore, reasoning about consensus instability requires further adaption of the chosen approach.

DeFi composability does not imply the security of contracts: The introduced notion of DeFi composability in [BDKJ21] is only concerned with the value that can be extracted by a given actor P . On the one hand, this definition of DeFi composability does not consider the state of contracts, which leads to a situation in which the deployment of another contract C' can trigger an existing contract C to reach a different (locked) state, or even self-destruct, such that C can no longer operate. This would not be a problem under the notion of DeFi composability. Suppose the value of the extractable funds does not increase significantly. In this case, C' and C are still DeFi composable, even if every player would lose significant amounts of funds if C' is deployed.

⁹cf. <https://solana.com/>.

¹⁰cf. <https://polygon.technology/>.

On the other hand, DeFi composability can be instantly violated by deploying a contract C' which hands out funds x , where $x > \varepsilon$, to the first player who calls the respective contract. In this case, DeFi composability is violated for contract C' and every player P , although only one player might be able to collect the promised (bribing) funds in the end.

Model does not consider forks: As also noted in the paper [BDKJ21] the model currently does not consider the likelihood and potential impact of forks in forkable blockchains.

Scope limited to a single system/domain/base resource: Currently, the model presented in the paper [BDKJ21] considers only balances of the primary asset and tokens/contracts based within the same system/domain i.e., smart contract cryptocurrency. The `balance()`[T] function is used to provide the balance of the respective asset T, conversions and exchange rate to the primary currency/asset are not discussed. Therefore, no cross-chain interactions and value extraction opportunities are modeled. This topic was addressed by another recent publication [OSS⁺21] which will be summarized and discussed in Section A.8.2.

A.8.2 Unity is Strength

In concurrent work [OSS⁺21] the authors address the concept of *cross-domain (maximal) extractable value* and outline a series of open questions in relation to this concept. Therefore, they adopt the definitions of EV, MEV/k-MEV given in [BDKJ21] and extend them to a cross-domain i.e., cross-chain context. The authors of [OSS⁺21] define extractable value as follows:

“Extractable value is the value between one or more blocks accessible to any user in a domain, given any arbitrary re-ordering, insertion and censorship of pending or existing transactions.”

More formally, they define the extractable value from the perspective of a user/player P , in a specific domain/system i as the difference in his overall balance after a sequence of actions a_1, \dots, a_n is executed on an initial state s . For a summary of relevant variables and symbols introduced in the respective paper [OSS⁺21], see Table A.5.

$$ev_i(P, s, a_1, \dots, a_n) = b_i(s', P) - b_i(s, P) \quad (\text{A.8})$$

Note that this definition is more abstract in the sense that it does not talk about blocks but actions and thus might be applicable to a broader range of systems compared to [BDKJ21]. Based on this definition of extractable value, the concept of *maximal extractable value* (MEV) is defined as follows:

“*Maximal Extractable Value (MEV) is the maximal value extractable between one or more blocks, given any arbitrary re-ordering, insertion or censorship of pending or existing transactions.*”

$$mev_i^j(P, s) = \max_{a_1, \dots, a_n \in A_j} (ev_i(P, s, a_1, \dots, a_n)) \quad (\text{A.9})$$

Here j is the domain in which the actions are taken, and i is the domain in which the balance change occurs. In a one-domain context $i = j$. Again, as in [BDKJ21] this concept refers to the *maximum value* that can be extracted, by a certain sequence of actions. The question, though is how can this sequence of actions be defined and then explored in practice? See Section A.8.2 regarding a discussion on this subject. The given definition of mev can then be extended to a cross-domain setting by adding a domain. Here is an example of a two-domain setting:

$$mev_{i,k}^{i,j}(P, s) = \max_{a_1, \dots, a_n \in A_i \cup A_j} (ev_i(P, s, a_1, \dots, a_n) + p_{i \rightarrow j}(ev_j(P, s, a_1, \dots, a_n))) \quad (\text{A.10})$$

This can, of course, be further extended by including more domains, a generalization for n domains is given in [OSS⁺21]. At the end the paper [OSS⁺21] closes with a series of open questions, which we will also discuss in Section A.8.2:

- I How do we best define the action space?
- II Aside from cross-domain arbitrage, what are other forms of cross-domain MEV?
- III What does a protocol for sequencer collusion look like and what are its desirable properties?
- IV How can we identify and quantify cross-chain MEV extraction taking place?
- V What can we learn from existing distributed and parallel programming literature?

Challenges, Open Problems and the Relation to this Paper

In this section, we outline the different definitions of MEV given in the paper [OSS⁺21] and their relation to the paper [JSSW22a]. In this context, we discuss some challenges and open problems that arise from the definitions of extractable value introduced in the paper [OSS⁺21].

Variable	Description
P	A player
s	A system state
$A_P(s)$	Set of all actions available to player P in state s
$a_1, \dots, a_n \in A_P$	Series of n actions of player P
A	Set of all actions of all players in the protocol
i, j	Two different domains / systems
$b_i(s, P)$	Functions returning the balance of a player
$p_{j \rightarrow i}$	Pricing function, indicating the balance in domain j is expressed in units of the native asset of domain i

Table A.5: Overview of important notation and variables used in the model presented in [OSS⁺21].

Definitions: In resemblance to the definitions given in paper [BDKJ21], the definition in [OSS⁺21] also uses the player and the current state as parameters to the function EV. In contrast to [BDKJ21], the other parameter is actions instead of blocks or transactions. This makes this definition more abstract and thus broader applicable. MEV is defined as the *maximum* extractable value from a sequence of actions, while the action space is not yet well defined and left to future work. Furthermore, cross-domain scenarios are considered, which would roughly resemble a multi-resources settings in the model of the paper [JSSW22a]. The analysis of the implications of such a cross-domain model is left to future work.

Sets of players and actions: The definition of the available actions space is stated as an open question in the paper [OSS⁺21]. The probabilistic nature of the interactions in Nakamoto-Style systems was already mentioned in the paper [OSS⁺21] as a factor that makes the definition of the available action space a challenging task.

We now want to add some more aspects, which introduce additional challenges. The combinatorial explosion of possible orderings of transactions as well as the required information, was already discussed in Section A.8.1, and can also be considered a challenge in defining the action space. Furthermore, at least in theory, current permissionless [Vuk15] Nakamoto-Style systems allow players (especially miners) to join and leave at any time. Therefore, the set of players is theoretically infinite. As a result, also the overall action space of players is theoretically unbounded as the number of players is unbounded.

In practice, given a finite world, the set of players is still challenging to define as it might change drastically over time. This, in turn, also changes the available action space over time.

Balance and Pricing of Assets: In the paper [OSS⁺21], a pricing function was assumed to model the exchange of units in different domains into the native asset of the base domain. Also, more complex models which depend on the supply and demand or the quantity exchanged are mentioned. In addition to this discussion on how to define the price of different assets in different domains, we would like to add the following challenge: The price or utility of a given asset is highly dependent on a specific player and his investment and interest in different systems/domains. Therefore, the price of the very same asset might be different for two different players due to the domains they are invested in or things such as colored coins [Ros12]. The pricing function also implicitly assumes perfect information regarding the exchange rates.

A.9 Proof that Finite Blocks Require Protocol Updates

As pointed out by Pass et al. [PS17], permissionless PoW blockchains cannot stop without becoming insecure, as equally long or even longer heavier chains can be constructed with a different history after the protocol has terminated. This would make it impossible for a new participant to distinguish a chain created in retrospect from the original discontinued longest chain. Thus, permissionless PoW blockchains are required to run infinitely long in theory. This implies that the individual block size cannot be static and finite, as outlined in the following proposition.

Proposition 3 (Finite blocks require protocol updates). A NC blockchain protocol that uses blocks that are finite in size requires protocol updates to increase the block size to provide a consistent chain of blocks, where every block has exactly one valid predecessor.

Proof. Since blocks are cryptographically linked to each other using a cryptographically secure hash function, the hash of the previous block requires space in the size of the output of the hash function, which is defined to be κ bits. Let's assume a blockchain that is only a concatenation of κ bit hash and a κ bit nonce value without any additional payload and a static difficulty. If the cryptographic hash function is modeled as a random oracle, the set of available outputs is exhausted as soon as the 2^κ blocks have been concatenated. Thus, the previous block hash of block $2^\kappa + 1$ leads to a collision, breaking the integrity and consistency¹¹ property of the chain. Therefore, a protocol update is required in block number 2^κ to increase the available space in a block and potentially the used hash function to one with a larger output space to avoid collisions. \square

As we restrict our model to only consider finite chain instances, the usage of finite blocks does not lead to the stated issue.

¹¹In comparison to the *backbone model* [GKL20], a consistent chain would require that no *insertions*, *copies*, or *predictions* of blocks are possible [GKL20].

List of Figures

2.1	Data structures and location of the hash placed in a Bitcoin block when Bitcoin is used as a parent chain in the context of merged mining. The <i>BlockHash / MerkleRoot</i> is pointing to the respective block(s) in the merge mined child chain(s).	21
2.2	Example of blocks mined after the hard-fork into two cryptocurrencies Π and $\Pi'u$	26
2.3	Hashrates of the defender/counter-attacker in the respective chains Π and Π' for different values of pitchfork attacker hashrate p_B . The colored orange square marks the area in which the counter-attackers \mathcal{A} loses in both chains, whereas the colored blue square marks the area in which the counter-attackers \mathcal{A} can retain the majority in both systems.	31
2.4	Visual comparison of hashrates in the two systems Π and Π' , before and after counter attacking the merge minded pitchfork protocol Π' , when merged mining of the parent is possible without negative effects on the parent chain. In this example, the hashrate of the attacker is $p_B = \frac{1}{3}$, which results in a situation where no player can clearly win on any of the two chains in case of a counter-attack.	32
2.5	Visual comparison of hashrates in the two systems Π and Π' , before and after counter attacking the merge minded pitchfork protocol Π' , when merged mining of the parent is not possible without negative side effects on the parent chain. In this example, the hashrate of the attacker is $p_B = \frac{1}{4}$, which results in a situation where no player can clearly win on any of the two chains in case of a counter-attack.	33
4.1	Example timeline showing blockchain structure and resulting payouts of a failed, and a successful tx revision attack with out-of-band payments. After k_V blocks on the target chain have passed, the attack contract is initialized with (at least) $k_V + 1$ block templates. The double-spend transaction(s) are included in block b'_1 . The payouts are performed in block e_T . The colored blocks are rewarded by the attack contract, either with their original value (reward + fee normalized to 1) or with an additional ϵ if the attack was successful. The numbers above colored blocks indicated those normalized rewards. If the attack succeeds, the first k_V blocks on the Bitcoin main chain also have to be compensated to provide an incentive for the respective miners (of those blocks) to also mine on the attack chain.	67

4.2	Attack success probability of a double-spending attack depending on the amount of blocks N that can be compensated/rewarded and different values for the rational hashrate $p_{\mathcal{R}}$. The number of required confirmation blocks by Vincent is set to $k_V = 6$	72
4.3	Example timeline showing blockchain structure and resulting payouts of a failed, and a successful transaction exclusion and ordering attack with out-of-band payments. The attack is initialized when the attack contract is published in block e_0 . Block templates are published as transactions in the funding cryptocurrency and refer to blocks in the target cryptocurrency. The payouts can be performed after k_B blocks. The colored blocks are rewarded by the attack contract, either only with their original value (reward + fee normalized to 1) or with an additional ϵ if the attack was successful. The numbers above colored blocks indicated those normalized rewards for the respective block.	82
5.1	The expected reward for the next block ρ is given on the y-axis, while the hashrate p_m of the miner under consideration is given on the x-axis. The figures shows a comparison of the profitability of mining on the main chain without any forks/attack and mining on a fork/attack with $p_B = p_E = p_V = 0$ and $z = 1$	100
5.2	The expected reward (ρ) for the next block is given on the y-axis, while the hashrate p_m of the miner under consideration is given on the x-axis. The figures shows a comparison of the profitability of different scenarios for mining on the main chain, versus mining on the attack chain.	102
5.3	Markov chain for modeling finite attacks and persistent victims (p_V). . .	104
5.4	Comparison of the infinite attack success probability to our finite Markov chain model for increasing attack duration $N = \vec{k}$ (x-axis).	105
5.5	Minimum bribe value per block ϵ , given in normalized block rewards, for different models and configurations. Thereby, ϵ reaches from 0 (blue) to 10 or higher (red). The y-axis shows p_E , while the x-axis shows p_m . The dashed line marks $p_E + p_m = 0.5$	107
6.1	Figure to approximate δ by comparing the average block rewards received by a miner with $p = 0.1$, to the expected infinite game rewards for the same miner with different discount factors δ . All rewards are given in normalized block rewards.	122
6.2	Visual illustration of different events and their consequences on a participant with $\mathcal{R} := \{R_0, R_1\}$. The total valuation of a participant <i>before</i> the respective event is normalized to 1. This means that the values for the initial exchange rates are $e_{0,1} = 1$ (s.t. $f(r_{0,1}, 1) = r_{0,1}$), and that δ is static and thus ignored ($\delta_{0,1} = 0$). In other words expected future rewards where already accounted for in the relation between $p_{0,1}$ and $r_{0,1}$, s.t. $p_0 + r_0 + p_1 + r_1 = 1$	128

6.3	Visual illustration of different events and their consequences on a participant with $\mathcal{R} := \{R_0, R_1\}$. The total valuation of a participant <i>before</i> the respective event is normalized to 1. This means that the values for the initial exchange rates are $e_{0,1} = 1$ (s.t. $f(r_{0,1}, 1) = r_{0,1}$), and that δ is static and thus ignored ($\delta_{0,1} = 0$). In other words expected future rewards were already accounted for in the relation between $p_{0,1}$ and $r_{0,1}$, s.t. $p_0 + r_0 + p_1 + r_1 = 1$	129
7.1	A blockchain is depicted as a specific sequence of n blocks within the set of all possible valid blocks. Starting from the latest block at height n , two potential next blocks in two different sets of valid future blocks are depicted. These two sets differ depending on the available set of transactions, i.e., $\vec{\mathcal{O}} = \text{FUTURE}(b_n, \mathcal{T})$ and $\vec{\mathcal{O}}' = \text{FUTURE}(b_n, \mathcal{T}')$, where $\mathcal{T} \subset \mathcal{T}'$ and thus $\vec{\mathcal{O}} \subset \vec{\mathcal{O}}'$	161
A.1	The figure shows an ongoing-, a failed- as well as a successful Transaction exclusion attack with in-band payments. The attack is initialized when the attack contract is published in block e_1 . If the unwanted transaction has been included, this can be proven to the attack contract, as shown in the failure case in block e_{N+x} . The payouts are performed in block e_T . The colored blocks are rewarded by the attack contract with an additional ϵ	211
A.2	Finite Markov chain for calculating the probability of mining at least ℓ consecutive blocks with hashrate $p_{\mathcal{R}}$	213
A.3	Minimum bribe value ϵ in normalized block rewards for different models and configurations.	219

List of Tables

2.1	Structure of the coinbase of a merge-mined block. Uses Namecoin as an example [Namb]	22
3.1	caption	46
3.2	Strategies to gain capacity in Nakamoto consensus according to [Bon18], augmented with AIM strategies (colored background).	55
4.1	Comparison of attack costs for $k_V = 6$, all costs given in BTC. The costs for the whale attack are the average from 10^6 simulation results provided in [LK17]. For comparison different Bitcoin block reward epochs (12.5 and 6.25 BTC) are provided for our P2W attack, all with $cost_{operational} = 0.5$ BTC, and average fee per block of 2 BTC and a bribe $\epsilon = 1$ BTC.	72
4.2	Overview of operational costs $c_{operational}$ for each of the main Ethereum smart contract operations of the attack contract (including the EMR) executing a successful attack on Bitcoin with $k_V = 6$ and $k_B = 6$	77
7.1	Overview of classes of protocol changes from old to new rules $\Pi \rightarrow \Pi'$. Thereby, \mathcal{O} and \mathcal{O}' denote the sets of valid blocks under the old and new protocol rules. The new set of valid blocks \mathcal{O}' is colored gray. \mathcal{U} denotes the validity set changes introduced by the protocol update.	146
7.2	Overview of different types of protocol compositions of two protocols Π (original) and Π' (new). Thereby, \mathcal{O} and \mathcal{O}' denote the sets of valid blocks under the different protocol rules.	148
A.1	Comparison of minimum bribe required per block ϵ for $\rho_{fork\ bl.} > \rho_{main\ bl.}$ with probabilities calculated using our Markov chain. The axis iterate the hashrate of an individual miner p_m , and other attackers $p_{\mathcal{E}}$. The table also shows the expected reward of miner m , if p_m would be directed towards the attack chain $\rho' = \rho_{fork\ bl.}$, as well as the expected reward $\rho = \rho_{main\ bl.}$, if p_m would be directed towards the main chain. All attacks start with a disadvantage of $z = 1$, a duration of $N = 8$ and the following configuration of the Markov chain $\vec{k} = 1, \overleftarrow{k} = 10, \eta_{attack} = 0, \eta_{main} = 0, p_V = 0.000000$	216
		233

A.2	Comparison of minimum bribe required per block ϵ for $\rho_{fork\ bl.} > \rho_{main\ bl.}$ with probabilities calculated using our Markov chain. The axis iterate the hashrate of an individual miner p_m , and other attackers $p_{\mathcal{E}}$. The table also shows the expected reward of miner m , if p_m would be directed towards the attack chain $\rho' = \rho_{fork\ bl.}$, as well as the expected reward $\rho = \rho_{main\ bl.}$, if p_m would be directed towards the main chain. All attacks start with a disadvantage of $z = 1$, a duration of $N = 8$ and the following configuration of the Markov chain $\vec{k} = 3, \overleftarrow{k} = 10, \eta_{attack} = 0, \eta_{main} = 1, p_{\mathcal{V}} = 0.100000$	217
A.3	Comparison of minimum bribe required per block ϵ for $\rho_{fork\ comp.} > \rho_{main\ comp.}$ with effort-related compensation and probabilities calculated using our Markov chain. The axis iterate the hashrate of an individual miner p_m , and other attackers $p_{\mathcal{E}}$. The table also shows the expected reward of miner m , if p_m would be directed towards the attack chain $\rho' = \rho_{fork\ comp.}$, as well as the expected reward $\rho = \rho_{main\ comp.}$, if p_m would be directed towards the main chain. All attacks start with a disadvantage of $z = 6$ and a duration of $N = 8$ and the following configuration of the Markov chain $\vec{k} = 3, \overleftarrow{k} = 10, \eta_{attack} = 0, \eta_{main} = 0, p_{\mathcal{V}} = 0.100000$	218
A.4	Overview of necessary notation and variables used in the model presented in [BDKJ21].	222
A.5	Overview of important notation and variables used in the model presented in [OSS ⁺ 21].	226

Notation, Variables and Symbols

Notation	Description
$:=$	Assignment operator e.g., $x := y$
$=$	Equality operator e.g., $x = x =$
$0x\text{ff}$	The prefix $0x$ denotes an hexadecimal representation. In this case the hexadecimal representation of the decimal number 255
$x[251 : 255]$	Denotes a range in the representation of variable x e.g., refers to the bits from 251 to 255 of variable x
$x y$	Denotes the concatenation of the strings x and y i.e., string concatenation
$ \mathcal{S} $	Cardinality of \mathcal{S} i.e., the number of elements in the set or sequence e.g., $ \{a, b, c\} = 3$
$[\mathcal{S}]^x$	Set of all x element subsets of set \mathcal{S} e.g., $\mathcal{S} := \{a, b, c\}$, $x := 2$ then this would be $\{\{a, b\}, \{a, c\}, \{b, c\}\}$, where $[\mathcal{S}]^x$ is equivalent to $\binom{ \mathcal{S} }{x}$. The total number of distinct subsets of a set \mathcal{S} is given by $\sum_{x=0}^{ \mathcal{S} } \binom{ \mathcal{S} }{x} = 2^{ \mathcal{S} }$
\cup, \cap, \setminus	Classical set operations: union, intersection, difference
\emptyset	The empty set
$s := (a, b, c)$	Sequence of ordered elements, such that a is before b and b is before c , i.e., $a \prec b \prec c$
$s := (s_1, s_2)$	Concatenation of two sequences s_1 and s_2 , where sequence s_2 is appended to s_1 resulting in the new sequence s
(\dots, \hat{x}_n)	Removal of the last element x_n in a sequence $s := (\dots, x_n)$, where $ s = n$ e.g., if $s := (s_1, s_2)$ then $(s_1, \hat{s}_2) = s_1 = s$
$s_0 \prec s_1$	Denotes that sequence s_0 is a prefix of sequence s_1
$s_0 \preceq s_1$	Denotes that sequence s_0 is either a prefix of sequence s_1 or equal to sequence s_1

Continued on next page

Table 1 – *Continued from previous page*

Notation	Description
$H(x)$	Generic cryptographic hash function. Takes a message x of arbitrary but finite size and outputs a fixed size hash h (also called digest).
$H^x(\cdot)$	Chained use of Hash function x times e.g., $H^2(x) = H(H(x))$
z	Number of leading <i>zero bits</i> of valid PoW hashes in Bitcoin i.e., of the 256 bit number T
\mathcal{P}	The set of participants/actors/players i.e., $\mathcal{P} = \{P_1, P_2, \dots, P_{ \mathcal{P} }\}$
P	Individual participant/actor/player
p	Voting power in a NC style system, which in terms of PoW cryptocurrency is measured in hashrate.
\mathcal{M}	The set of miners, i.e., players with voting p in the system e.g., hashrate
\mathcal{U}	The set of users, i.e., players without voting p in the system e.g., hashrate
\mathcal{B}	The set of byzantine participants/actors/players encompassing all palyers that collude towards a common objective
B	Individual byzantine participant/actor/player
\mathcal{A}	The set of altruistic participants/actors/players following the rules of the protocol
A	Individual altruistic participant/actor/player
\mathcal{R}	The set of economically rational participants/actors/players following the rules of the protocol as long as it is the most profitable option available
R	Individual altruistic participant/actor/player
\mathcal{V}	The set of participants/actors/players that suffers value loss from a particular attack i.e., the victims
V	Individual participant/actor/player that is a victim of an attack
E	Individual participant/actor/player that launches an attack to exploit some opportunity which results in higher profit
I	Individual participant/actor/player that indifferent to an ongoing attack as long as it has no consequences on his individual profits

Continued on next page

Table 1 – Continued from previous page

Notation	Description
o	A abstraction state transition, sometimes also referred to as <i>operation</i> e.g., a block b in the context of most cryptocurrencies
\mathcal{O}	The set of all possible valid operations/blocks starting from a given genesis block b_1
$\vec{\mathcal{O}}$	The set of all possible reachable valid operations from a given starting state.
\mathcal{D}	The set of all possible inputs to $\text{VALID}(\cdot)$ including invalid ones
\mathcal{S}	The set of all potentially reachable valid system states.
tx	A transaction i.e., smallest state transition unit in the system. If used, the subscript tx_x refers to a specific transaction at position x
τ	Ordered sequence of transactions, e.g., (tx_0, tx_1, tx_2, tx_3)
\mathcal{T}	A unordered set of transactions, e.g., $\{tx_3, tx_1\}$, like the transaction pool. If further superscript and subscript is used, this indicates the set of all known transactions by player i , till a given block height, including the unconfirmed transaction pool e.g., \mathcal{T}_n^i
\aleph	Transaction limit, which refers the maximum number of transactions which can be included in one batch/operation e.g., a block
b	A block (i.e., a state transition) for some blockchain. If used, the subscript $b_{xy}(\cdot)$ refers to a block mined at position x in the chain e.g., b_1 is the genesis block
\bar{b}	A block template, which is a block that is defined in terms of its transactions, i.e., state transitions, but not yet mined such that it contains a valid PoW
c	A chain of blocks linked with cryptographic hashes, i.e., a blockchain. If used, the subscript gives the length of the chain in terms of its blocks. A blockchain represents a certain sequence of transactions s , e.g., $c_1 = (b_1(tx_0), b_2(tx_1), b_3(tx_2, tx_3)), c_2 = (b_1(tx_0), b_2(tx_1), b'_3(tx_2))$
\bar{c}	A blockchain template \bar{c}_x which is a chain of blocks with a suffix of one or multiple blocks, which have already been specified regarding their content, but do not yet include a valid PoW.

Continued on next page

Table 1 – *Continued from previous page*

Notation	Description
\mathcal{C}	The set of all potentially constructable valid blockchains starting from an initially defined genesis block b_1 defined by the protocol rules.
DAG	Directly Acyclic Graph (DAG) of blocks, encompassing multiple chains.
$InTree$	In-Tree (also called anti-arborescence) where blocks are vertices and a edges are cryptographic hash points to the previous block. The resulting graph represents an in-tree encompassing multiple chains.
k	Global security parameter, specifying the number of suffix blocks to discard s.t. the rest can be considered the <i>common prefix</i> [GKL20, SZ16], of the chain, which is the same for all honest nodes
κ	Security parameter for required cryptographic primitives, e.g., in [GKL20] κ specifies the length of the hash function
N	Duration of an attack in blocks, i.e., the maximum number of blocks/steps the attack can be sustained
z	The number of blocks the attack is behind a competing chain when the attack starts
\overleftarrow{k}	The number of blocks an attack chain can fall behind s.t. the attack is assumed to clearly lose
\overrightarrow{k}	The number of blocks an attack chain is before a competing chain s.t. the attack is assumed to clearly win
η	The number of contributed blocks to an chain during a fork/attack
r	Amount of a certain resource the respective player has.
e	Personal exchange rate of a certain resource the respective player has.
δ	Personal discount factor for a certain resource the respective player has.
R	A single resource, defined by the by a quadruple $R_i = \langle r, e, p, \delta \rangle$
\mathcal{R}	Set of all resources an individual player cares about, the respective player might be indicated by a subscript \mathcal{R}_i .
F	Overall available quantity of a certain resource.
$cost$	Some cost value for the event given as subscript.

Continued on next page

Table 1 – *Continued from previous page*

Notation	Description
rst	An optional result of the execute function
$\text{EXECUTE}(s, o)$	The execute function taking a state (s) and an operation (o), and returning a new state s' and optionally a result rst . In case of most Nakamoto consensus blockchains, there is no dedicated result, just a state like for example the state tree in Ethereum, or the UTXO set.
$\text{EEXECUTE}(s, o)$	An alternative version of the execute function, taking the same argument, but returning \perp as a response on more inputs than the original execute function.
REACHABLE	Predicate which returns true if a desired operation is reachable given a set of available transactions as well as a starting operation
FUTURE	Returns the set of future reachable operations, given a starting operations and a set of available transactions
$\text{LENGTH}(c)$	Return the length of a given chain in blocks. Same as $ c = \text{LENGTH}(c)$.
$\text{HEAD}(c)$	Returns the last block at the end of the given chain.
TXSEQ	Returns the ordered sequence of transactions from the given block or chain
$\text{CHAIN}(b)$	Returns the entire ordered sequence of blocks (i.e., the blockchain) leading to b which determined all predecessors in this chain
ORDER	Orders the set or sequence of transactions in some way.
FUNDS	Function that return the funds of an individual player or set of players for a given state, resource/cryptocurrency and exchange rate. Hereby, the state can be the height of a blockchain. It is also possible, that state, resource/cryptocurrency and exchange is omitted if clear from the context e.g., $\text{FUNDS}(B)$
\mathbb{P}	Probability of some event i.e., $0 \leq \mathbb{P}(x) \leq 1$
ρ	The profit for some player i , after subtracting his costs i.e., ρ_i
$\text{VALID}(c)$	Predicate that returns true if a given sequence of operations e.g., a chain of blocks, leads to a valid system state, false otherwise. This predicate invokes the function $\text{EXECUTE}(\cdot)$.

Continued on next page

Table 1 – *Continued from previous page*

Notation	Description
EVALID(c)	Predicate that returns true if a given sequence of operations e.g., a chain of blocks, leads to a system state considered valid by some attacker, false otherwise. This predicate invokes the function EEXECUTE(\cdot).
WELLFORMED(tx)	Predicate that returns true if a given transaction or set of transactions is well-formed i.e., there exists a valid system state in which the given transaction is valid

Bibliography

- [AAC⁺05] Amitanand S Aiyer, Lorenzo Alvisi, Allen Clement, Mike Dahlin, Jean-Philippe Martin, and Carl Porth. BAR fault tolerance for cooperative services. In *ACM SIGOPS operating systems review*, volume 39, pages 45–58. ACM, 2005. Issue: 5.
- [BBH⁺12] Jörg Becker, Dominic Breuker, Tobias Heide, Justus Holler, Hans Peter Rauer, and Rainer Böhme. Can we afford integrity by proof-of-work? Scenarios inspired by the Bitcoin currency. In *WEIS*. Springer, 2012.
- [BBH⁺13] Jörg Becker, Dominic Breuker, Tobias Heide, Justus Holler, Hans Peter Rauer, and Rainer Böhme. Can We Afford Integrity by Proof-of-Work? Scenarios Inspired by the Bitcoin Currency. In Rainer Böhme, editor, *The Economics of Information Security and Privacy*, pages 135–156. Springer, 2013.
- [BBTL22] George Bissias, Rainer Böhme, David Thibodeau, and Brian N. Levine. Pricing Security in Proof-of-Work Systems. *The 2022 Workshop on the Economics of Information Security, 2022*.
- [BCD⁺14] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains. Technical report, 2014.
- [BDKJ21] Kushal Babel, Philip Daian, Mahimna Kelkar, and Ari Juels. Clockwork Finance: Automated Analysis of Economic Security in Smart Contracts. Technical Report 1147, 2021.
- [BGM⁺18] Christian Badertscher, Juan Garay, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. But Why Does It Work? A Rational Protocol Design Treatment of Bitcoin. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, Lecture Notes in Computer Science, pages 34–65, Cham, 2018. Springer International Publishing.
- [Bita] Bitcoin community. Bitcoin developer guide. <https://developer.bitcoin.org/devguide/>. Accessed: 2021-01-31.

- [Bitb] Bitcoin community. Bitcoin wiki. <https://bitcoin.it/>. Accessed: 2015-06-30.
- [BMC⁺15] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In *IEEE Symposium on Security and Privacy*, 2015.
- [BMTZ17] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Bitcoin as a Transaction Ledger: A Composable Treatment. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, Lecture Notes in Computer Science, pages 324–356, Cham, 2017. Springer International Publishing.
- [Böh19] Rainer Böhme. Talk: A primer on economics for cryptocurrencies. <https://bdlt.school/files/slides/talk-rainer-b%C3%B6hme-a-primer-on-economics-for-cryptocurrencies.pdf>, 2019. Accessed: 2020-09-15.
- [Bon16] Joseph Bonneau. Why buy when you can rent? Bribery attacks on Bitcoin consensus. In *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, 2016.
- [Bon18] Joseph Bonneau. Hostile blockchain takeovers (short paper). In *Financial Cryptography and Data Security - FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curacao, March 2, 2018, Revised Selected Papers*, 2018.
- [btg20] Bitcoin gold (btg) was 51% attacked. <https://gist.github.com/metalicjames/71321570a105940529e709651d0a9765>, 2020. Accessed: 2020-09-15.
- [Bud18] Eric Budish. The economic limits of bitcoin and the blockchain. In *NBER Working Paper Series*. National Bureau of Economic Research, 2018.
- [But16] Vitalik Buterin. Chain Interoperability. <https://allquantor.at/blockchainbib/pdf/vitalik2016chain.pdf>, 2016. Accessed: 2016-09-09.
- [But17] Vitalik Buterin. Hard forks, soft forks, defaults and coercion. http://vitalik.ca/general/2017/03/14/forks_and_markets.html, 2017. Accessed: 2017-04-11.
- [But21] Vitalik Buterin. Two-slot proposer/builder separation. <https://ethresear.ch/t/two-slot-proposer-builder-separation/10980>, 2021. Accessed: 2022-11-14.

- [Can00] Ran Canetti. *Universally Composable Security: A New Paradigm for Cryptographic Protocols*. 2000. Published: Cryptology ePrint Archive, Report 2000/067.
- [CDE⁺16] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, and Emin Gün. On Scaling Decentralized Blockchains. In *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, 2016.
- [CDFZ17] Alexander Chepurnoy, Tuyet Duong, Lei Fan, and Hong-Sheng Zhou. TwinsCoin: A Cryptocurrency via Proof-of-Work and Proof-of-Stake. In *Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts, BCC@AsiaCCS 2018, Incheon, Republic of Korea, June 4, 2018*, 2017.
- [CKWN16] Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 154–167. ACM, 2016.
- [CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, 2001. The Knuth-Morris-Pratt Algorithm.
- [CM12] Jing Chen and Silvio Micali. Collusive dominant-strategy truthfulness. *J. Econ. Theory*, 147(3):1300–1312, 2012.
- [Coi] CoinMarketCap. Coinmarketcap: Cryptocurrency market capitalizations. <https://coinmarketcap.com/>. Accessed 2021-01-12.
- [coi19a] coindesk. Bitcoin cash miners undo attacker’s transactions with 51% attack?. <https://www.coindesk.com/bitcoin-cash-miners-undo-attackers-transactions-with-51-attack>, 2019. Accessed: 2020-09-15.
- [coi19b] cointelegraph. Ethereum classic 51% attack — the reality of proof-of-work. <https://cointelegraph.com/news/ethereum-classic-51-attack-the-reality-of-proof-of-work>, 2019. Accessed: 2020-09-15.
- [coi20] coindesk. Ethereum classic suffers second 51% attack in a week. <https://www.coindesk.com/ethereum-classic-suffers-second-51-attack-in-a-week>, 2020. Accessed: 2020-09-15.
- [CSV16] Christian Cachin, Simon Schubert, and Marko Vukolic. Non-determinism in Byzantine Fault-Tolerant Replication. In *20th International Conference on Principles of Distributed Systems, OPODIS 2016, December 13-16, 2016, Madrid, Spain*, LIPIcs, 2016.

- [CT96] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. In *Journal of the ACM (JACM)*, volume 43, pages 225–267. ACM, 1996. Issue: 2.
- [Cun] Cunicula. Bribery: The double double spend. Bitcoin Forum <https://bitcointalk.org/index.php?topic=122291>. Accessed: 2021-1-31.
- [DGK⁺20] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 910–927. IEEE, 2020.
- [DKT⁺20] Amir Dembo, Sreeram Kannan, Ertem Nusret Tas, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. Everything is a Race and Nakamoto Always Wins. In *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 859–878. ACM, 2020.
- [DN92] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*, pages 139–147. Springer, 1992.
- [DW15] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*, pages 3–18. Springer, 2015.
- [EGSvR16] Ittay Eyal, Adem Efe Gencer, Emin Gun Sirer, and Robbert van Renesse. Bitcoin-NG: A Scalable Blockchain Protocol. In *13th USENIX Security Symposium on Networked Systems Design and Implementation (NSDI'16)*. USENIX Association, March 2016.
- [EMC19] Shayan Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. SoK: Transparent Dishonesty: front-running attacks on Blockchain. In *Financial Cryptography and Data Security - FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers*, pages 170–189. Springer, 2019.
- [ES14] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*, Lecture Notes in Computer Science, pages 436–454. Springer, 2014.
- [Eya15] Ittay Eyal. The miner’s dilemma. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 89–103. IEEE, 2015.

- [FB19] Bryan Ford and Rainer Böhme. Rationality is Self-Defeating in Permissionless Systems. Technical report, 2019. eprint: arXiv:1910.08820.
- [fom18] How the winner got the fomo3d prize - a detailed explanation. <https://medium.com/coinmonks/how-the-winner-got-fomo3d-prize-a-detailed-explanation-b30a69b7813f>, 2018. Accessed: 2020-09-15.
- [GBI18] Artur Gorokh, Siddhartha Banerjee, and Krishnamurthy Iyer. When Bribes are Harmless: The Power and Limits of Collusion-Resilient Mechanism Design. *SSRN preprint SSRN_id:3125003*, 2018.
- [GCR16] Ilias Giechaskiel, Cas Cremers, and Kasper B. Rasmussen. On Bitcoin Security in the Presence of Broken Cryptographic Primitives. In *European Symposium on Research in Computer Security (ESORICS)*, September 2016.
- [GKJ⁺21] Simin Ghesmati, Andreas Kern, **Aljosha Judmayer**, Nicholas Stifter, and Edgar R. Weippl. Unnecessary Input Heuristics and PayJoin Transactions. In *HCI International 2021 - Posters*, July 2021.
- [GKL14] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin Backbone Protocol: Analysis and Applications. Technical Report 765, 2014.
- [GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology-EUROCRYPT 2015*, pages 281–310. Springer, 2015.
- [GKL16] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. *The Bitcoin Backbone Protocol with Chains of Variable Difficulty*. 2016. Publication Title: IACR Cryptology ePrint Archive, Report 2016/1048.
- [GKL20] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. *The Bitcoin Backbone Protocol: Analysis and Applications*. 2020. Publication Title: IACR Cryptology ePrint Archive, Report 2014/765.
- [GKR20] Peter Gaži, Aggelos Kiayias, and Alexander Russell. Tight Consistency Bounds for Bitcoin. In *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*. ACM, 2020.
- [GKW⁺16] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC*, pages 3–16. ACM, 2016.
- [GL79] Jerry Green and Jean-Jacques Laffont. On coalition incentive compatibility. *The Review of Economic Studies*, 46(2):243–254, 1979. Publisher: JSTOR.

- [HBG16] Ethan Heilman, Foteini Baldimtsi, and Sharon Goldberg. *Blindly Signed Contracts: Anonymous On-Blockchain and Off-Blockchain Bitcoin Transactions*. 2016. Published: Cryptology ePrint Archive, Report 2016/056.
- [HC21] Binbing Hou and Feng Chen. A Study on Nine Years of Bitcoin Transactions: Understanding Real-world Behaviors of Bitcoin Miners and Users. 2021.
- [Her18] Maurice Herlihy. *Atomic Cross-Chain Swaps*. 2018. Published: arXiv:1801.09515.
- [Hom] Homepage. Tornado cash. <https://tornado.cash/>. Accessed: 2020-10-15.
- [Jam14] Harvey S. James. incentive compatibility. Encyclopedia Britannica, 2014. Accessed on 2021-01-19.
- [JKK14] **Aljosha Judmayer**, Lukas Krammer, and Wolfgang Kastner. On the security of security extensions for IP-based KNX networks. In Maria Gradinariu Potop-Butucaru and Hervé Rivano, editors, *10th IEEE International Workshop on Factory Communication Systems (WFCS 2014)*, Toulouse, France, May 2014.
- [JSKW17] **Aljosha Judmayer**, Nicholas Stifter, Katharina Krombholz, and Edgar Weippl. Blocks and Chains: Introduction to Bitcoin, Cryptocurrencies, and Their Consensus Mechanisms. *Synthesis Lectures on Information Security, Privacy, and Trust*, 9(1):1–123, 2017.
- [JSS20] **Aljosha Judmayer**, Philipp Schindler, and Nicholas Stifter. Blockchain-Technologie – Technische Erklärung. Technical report, January 2020. Published: LexisNexis.
- [JSSW18] **Aljosha Judmayer**, Nicholas Stifter, Philipp Schindler, and Edgar R. Weippl. Pitchforks in Cryptocurrencies: Enforcing Rule Changes Through Offensive Forking- and Consensus Techniques (Short Paper). In Joaquín García-Alfaro, Jordi Herrera-Joancomartí, Giovanni Livraga, and Ruben Rios, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2018 International Workshops, DPM 2018 and CBT 2018, Barcelona, Spain, September 6-7, 2018, Proceedings*, volume 11025 of *Lecture Notes in Computer Science*, pages 197–206. Springer, 2018.
- [JSSW22a] **Aljosha Judmayer**, Nicholas Stifter, Philipp Schindler, and Edgar Weippl. Estimating (Miner) Extractable Value is Hard, Let’s Go Shopping! In *Financial Cryptography and Data Security. International Workshops - CoDecFin*, 2022.
- [JSSW22b] **Aljosha Judmayer**, Nicholas Stifter, Philipp Schindler, and Edgar Weippl. How much is the fork? Fast Probability and Profitability Calculation during

Temporary Forks. In *1st International Cryptoasset Analytics Workshop (CAAW) co-located with the International World Wide Web Conference (WWW) 2022*, 2022.

- [JSZ⁺19] **Aljosh Judmayer**, Nicholas Stifter, Alexei Zamyatin, Itay Tsabary, Ittay Eyal, Peter Gazi, Sarah Meiklejohn, and Edgar Weippl. Pay To Win: Cheap, Crowdfundable, Cross-chain Algorithmic Incentive Manipulation Attacks on PoW Cryptocurrencies. Technical report, 2019.
- [JSZ⁺21a] **Aljosh Judmayer**, Nicholas Stifter, Alexei Zamyatin, Itay Tsabary, Ittay Eyal, Peter Gazi, Sarah Meiklejohn, and Edgar R. Weippl. Pay to Win: Cheap, Cross-Chain Bribing Attacks on PoW Cryptocurrencies. In Matthew Bernhard, Andrea Bracciali, Lewis Gudgeon, Thomas Haines, Arian Klages-Mundt, Shin'ichiro Matsuo, Daniel Perez, Massimiliano Sala, and Sam Werner, editors, *Financial Cryptography and Data Security. FC 2021 International Workshops - CoDecFin, DeFi, VOTING, and WTSC, Virtual Event, March 5, 2021, Revised Selected Papers*, volume 12676 of *Lecture Notes in Computer Science*, pages 533–549. Springer, 2021.
- [JSZ⁺21b] **Aljosh Judmayer**, Nicholas Stifter, Alexei Zamyatin, Itay Tsabary, Ittay Eyal, Peter Gazi, Sarah Meiklejohn, and Edgar R. Weippl. SoK: Algorithmic Incentive Manipulation Attacks on Permissionless PoW Cryptocurrencies. In Matthew Bernhard, Andrea Bracciali, Lewis Gudgeon, Thomas Haines, Arian Klages-Mundt, Shin'ichiro Matsuo, Daniel Perez, Massimiliano Sala, and Sam Werner, editors, *Financial Cryptography and Data Security. FC 2021 International Workshops - CoDecFin, DeFi, VOTING, and WTSC, Virtual Event, March 5, 2021, Revised Selected Papers*, volume 12676 of *Lecture Notes in Computer Science*, pages 507–532. Springer, 2021.
- [JUM⁺17] **Aljosh Judmayer**, Johanna Ullrich, Georg Merzdovnik, Artemios G. Voyiatzis, and Edgar R. Weippl. Lightweight Address Hopping for Defending the IPv6 IoT. In *12th International Conference on Availability, Reliability and Security (ARES)*, 2017.
- [JW16a] **Aljosh Judmayer** and Edgar R. Weippl. Condensed Cryptographic Currencies Crash Course (C5): Tutorial. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1857–1858, 2016.
- [JW16b] **Aljosh Judmayer** and Edgar R. Weippl. Cryptographic Currencies Crash Course (C4): Tutorial. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11-15, 2016, Companion Volume*, pages 1021–1024, 2016.
- [JZS⁺17] **Aljosh Judmayer**, Alexei Zamyatin, Nicholas Stifter, Artemios G. Voyiatzis, and Edgar R. Weippl. Merged Mining: Curse or Cure? In Joaquín

García-Alfaro, Guillermo Navarro-Arribas, Hannes Hartenstein, and Jordi Herrera-Joancomartí, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2017 International Workshops, DPM 2017 and CBT 2017, Oslo, Norway, September 14-15, 2017, Proceedings*, volume 10436 of *Lecture Notes in Computer Science*, pages 316–333. Springer, 2017.

- [JZSW17] **Aljosha Judmayer**, Alexei Zamyatin, Nicholas Stifter, and Edgar R. Weippl. Bitcoin - Cryptocurrencies and Alternative Applications. *ERCIM News*, 2017(110), 2017.
- [KDF13] Joshua A Kroll, Ian C Davey, and Edward W Felten. The economics of Bitcoin mining, or Bitcoin in the presence of adversaries. In *The 2013 Workshop on the Economics of Information Security*, 2013.
- [KF19] Shea Ketsdever and Michael J. Fischer. *Incentives Don't Solve Blockchain's Problems*. 2019. _eprint: arXiv:1905.04792.
- [KGDS18] Sukrit Kalra, Seep Goel, Mohan Dhawan, and Subodh Sharma. ZEUS: Analyzing Safety of Smart Contracts. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.
- [KJGW16] Katharina Krombholz, **Aljosha Judmayer**, Matthias Gusenbauer, and Edgar R. Weippl. The Other Side of the Coin: User Experiences with Bitcoin Security and Privacy. In *International Conference on Financial Cryptography and Data Security (FC)*, 2016.
- [KKKT16] Aggelos Kiayias, Elias Koutsoupias, Maria Kyropoulou, and Yiannis Tseleounis. Blockchain Mining Games. In *Proceedings of the 2016 ACM Conference on Economics and Computation, EC'16, Maastricht, The Netherlands, July 24-28, 2016*, 2016. Publication Title: CoRR.
- [KMZ17] Aggelos Kiayias, Andrew Miller, and Dionysis Zindros. *Non-interactive proofs of proof-of-work*. 2017. Published: Cryptology ePrint Archive, Report 2017/963.
- [KNS⁺18] Aashish Kolluri, Ivica Nikolic, Ilya Sergey, Aquinas Hobor, and Prateek Saxena. *Exploiting The Laws of Order in Smart Contracts*. 2018. Published: arXiv:1810.11605.
- [KNW20] Majid Khabbazian, Tejaswi Nadahalli, and Roger Wattenhofer. Timelocked Bribes. Technical Report 774, 2020.
- [Kur20] Klaus Kursawe. Wendy, the good little fairness widget. *IACR Cryptol. ePrint Arch.*, 2020:885, 2020.

- [KZGJ20] Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. Order-fairness for byzantine consensus. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 451–480. Springer, 2020.
- [LCW⁺06] Harry C Li, Allen Clement, Edmund L Wong, Jeff Napper, Indrajit Roy, Lorenzo Alvisi, and Michael Dahlin. BAR gossip. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 191–204. USENIX Association, 2006.
- [Ler] Sergio Demian Lerner. The bitcoin eternal choice for the dark side attack (ecdsa). <https://bitslog.com/2013/06/26/the-bitcoin-eternal-choice-for-the-dark-side-attack-ecdsa/>. Accessed: 2021-1-31.
- [LK17] Kevin Liao and Jonathan Katz. Incentivizing blockchain forks via whale transactions. In *International Conference on Financial Cryptography and Data Security*, pages 264–279. Springer, 2017.
- [LLR04] Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin. On the Composition of Authenticated Byzantine Agreement. Technical Report 181, 2004.
- [LLW] Eric Lombrozo, Johnson Lau, and Peter Wuille. Bitcoin improvement proposal 141 (bip141): Segregated witness. <https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki>. Accessed: 2018-06-28.
- [LVTS17] Loi Luu, Yaron Velner, Jason Teutsch, and Prateek Saxena. SMART POOL : Practical Decentralized Pooled Mining. In *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, pages 1409–1426. USENIX Association, 2017. Publication Title: Cryptology ePrint Archive, Report 2017/019.
- [MBKM19] Andrew Miller, Iddo Bentov, Ranjit Kumaresan, and Patrick McCorry. Sprites: Payment Channels that Go Faster than Lightning. In *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers*, pages 508–526. Springer, 2019. Publication Title: arXiv preprint arXiv:1702.05812.
- [MHM17] Patrick McCorry, Ethan Heilman, and Andrew Miller. Atomically Trading with Roger: Gambling on the Success of a Hardfork. In Joaquín García-Alfaro, Guillermo Navarro-Arribas, Hannes Hartenstein, and Jordi Herrera-Joancomartí, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2017 International Workshops, DPM*

2017 and CBT 2017, Oslo, Norway, September 14-15, 2017, Proceedings, volume 10436 of *Lecture Notes in Computer Science*, pages 334–353. Springer, 2017.

- [MHM18] Patrick McCorry, Alexander Hicks, and Sarah Meiklejohn. Smart Contracts for Bribing Miners. In *5th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security 18 (FC)*. Springer, 2018.
- [MJP⁺20] Michael Mirkin, Yan Ji, Jonathan Pang, Ariah Klages-Mundt, Ittay Eyal, and Ari Juels. BDoS: Blockchain Denial of Service. *Proceedings of the 2020 ACM SIGSAC conference on Computer and Communications Security*, January 2020. arXiv: 1912.07497.
- [MJVW17] Georg Merzdovnik, **Aljosha Judmayer**, Artemios G. Voyiatzis, and Edgar Weippl. A performance assessment of network address shuffling in IoT systems (Extended Abstract). 2017.
- [MM17] Sarah Meiklejohn and Rebekah Mercer. Möbius: Trustless Tumbling for Transaction Privacy. In *Proc. Priv. Enhancing Technol.*, pages 105–121, 2017. Published: Cryptology ePrint Archive, Report 2017/881.
- [Nak08] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. December 2008.
- [Nama] Namecoin. Homepage. <https://www.namecoin.org/>. Accessed: 2020-09-15.
- [Namb] Namecoin community. Bitcoin wiki - merged mining. https://en.bitcoin.it/wiki/Merged_mining_specification. Accessed: 2022-04-09.
- [NBF⁺16] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.
- [NKMS16] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *1st IEEE European Symposium on Security and Privacy, 2016*. IEEE, 2016.
- [NR99] Noam Nisan and Amir Ronen. Algorithmic mechanism design. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 129–140, 1999.
- [NR01] Noam Nisan and Amir Ronen. Algorithmic Mechanism Design. *Games Econ. Behav.*, 35(1-2):166–196, 2001.
- [Ope] Openethereum. Replace by fee in openethereum. <https://openethereum.github.io/Transactions-Queue.html>. Accessed: 2020-12-23.

- [OSS⁺21] Alexandre Obadia, Alejo Salles, Lakshman Sankar, Tarun Chitra, Vaibhav Chellani, and Philip Daian. Unity is Strength: A Formalization of Cross-Domain Maximal Extractable Value. *arXiv:2112.01472 [cs]*, December 2021. arXiv: 2112.01472.
- [P2P] P2Pool. P2pool. <http://p2pool.in/>. Accessed: 2022-06-28.
- [PD16] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network. Technical report, 2016.
- [PS17] Rafael Pass and Elaine Shi. Hybrid Consensus: Efficient Consensus in the Permissionless Model. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, LIPIcs, 2017.
- [PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the Blockchain Protocol in Asynchronous Networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, Lecture Notes in Computer Science, pages 643–673, Cham, 2017. Springer International Publishing.
- [QZG22] Kaihua Qin, Liyi Zhou, and Arthur Gervais. Quantifying Blockchain Extractable Value: How dark is the forest? In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pages 198–214. IEEE, 2022.
- [Rel] BTC Relay. <https://github.com/ethereum/btcrelay>. Accessed 2018-04-17.
- [RJZH19] Matteo Romiti, **Aljosha Judmayer**, Alexei Zamyatin, and Bernhard Haslhofer. A Deep Dive into Bitcoin Mining Pools: An Empirical Analysis of Mining Shares. In *The 2019 Workshop on the Economics of Information Security*, 2019.
- [RMSK14] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. CoinShuffle: Practical decentralized coin mixing for Bitcoin. In *Computer Security-ESORICS 2014*, pages 345–364. Springer, 2014.
- [Ros12] Meni Rosenfeld. *Overview of colored coins*. 2012. Publication Title: White paper, bitcoil.co.il.
- [Ros14] M. Rosenfeld. *Analysis of Hashrate-Based Double Spending*, volume abs/1402.2009. 2014. Publication Title: CoRR.
- [Ros17] Grigore Rosu. K: A semantic framework for programming languages and formal analysis tools. *Dependable Software Systems Engineering*, 50:186, 2017.

- [Ros21] Andreas Rosegger. *Race to the Door-A Goldfinger Attack on Proof of Work Cryptocurrencies*. PhD Thesis, Wien, 2021.
- [sAM] socrates1024 (Andrew Miller). Feather-forks: enforcing a blacklist with sub-50 <https://bitcointalk.org/index.php?topic=312668>. Accessed: 2021-1-31.
- [SAW19] Nicholas Stifter, Aljosha Aljosha, and Edgar R. Weippl. Revisiting Practical Byzantine Fault Tolerance Through Blockchain Technologies. In Stefan Biffl, Matthias Eckhart, Arndt Lüder, and Edgar R. Weippl, editors, *Security and Quality in Cyber-Physical Systems Engineering, With Forewords by Robert M. Lee and Tom Gillb*, pages 471–495. Springer, 2019.
- [Sch00] James Schummer. Manipulation through Bribes. *J. Econ. Theory*, 91(2):180–198, 2000.
- [SJH⁺21] Philipp Schindler, **Aljosha Judmayer**, Markus Hittmeir, Nicholas Stifter, and Edgar R. Weippl. RandRunner: Distributed Randomness from Trapdoor VDFs with Strong Uniqueness. In *Network and Distributed System Security Symposium 2021*, February 2021.
- [SJS⁺18] Nicholas Stifter, **Aljosha Judmayer**, Philipp Schindler, Alexei Zamyatin, and Edgar Weippl. *Agreement with Satoshi - On the Formalization of Nakamoto Consensus*. 2018. Published: Cryptology ePrint Archive, Report 2018/400.
- [SJS⁺21] Nicholas Stifter, **Aljosha Judmayer**, Philipp Schindler, Andreas Kern, and Walid Fdhila. *What is Meant by Permissionless Blockchains?* 2021. Published: Cryptology ePrint Archive, Report 2021/023.
- [SJSW20] Philipp Schindler, **Aljosha Judmayer**, Nicholas Stifter, and Edgar Weippl. HydRand: Practical Continuous Distributed Randomness. In *Proceedings of IEEE Symposium on Security and Privacy (IEEE S&P)*. IEEE, 2020.
- [SJSW22] Nicholas Stifter, **Aljosha Judmayer**, Philipp Schindler, and Edgar Weippl. Opportunistic Algorithmic Double-Spending: How I learned to stop worrying and hedge the Fork. In *ESORICS*, 2022.
- [SJW17] Nicholas Stifter, **Aljosha Judmayer**, and Edgar R. Weippl. A Holistic Approach to Smart Contract Security. *ERCIM News*, 2017(110), 2017.
- [SKH18] Ilya Sergey, Amrit Kumar, and Aquinas Hobor. Temporal Properties of Smart Contracts. In *Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice - 8th International Symposium, ISoLA 2018, Limassol, Cyprus, November 5-9, 2018, Proceedings, Part IV*, pages 323–338, 2018.

- [SSJ⁺19] Nicholas Stifter, Philipp Schindler, **Aljosha Judmayer**, Alexei Zamyatin, Andreas Kern, and Edgar Weippl. Echoes of the Past: Recovering Blockchain Metrics From Merged Mining. In *Proceedings of the 23rd International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2019.
- [SSJ21] Nicholas Stifter, Philipp Schindler, and **Aljosha Judmayer**. Blockchain-Technologie – Anwendungsformen. Technical report, April 2021. Published: LexisNexis.
- [SSZ16] Ayelet Sapirshstein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.
- [SZ16] Yonatan Sompolinsky and Aviv Zohar. Bitcoin’s Security Model Revisited. *arXiv preprint arXiv:1605.09193*, 2016.
- [TE18] Itay Tsabary and Ittay Eyal. The gap game. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 713–728. ACM, 2018.
- [TIGS21] Christof Ferreira Torres, Antonio Ken Iannillo, Arthur Gervais, and Radu State. The Eye of Horus: Spotting and Analyzing Attacks on Ethereum Smart Contracts. In *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part I*, pages 33–52. Springer, 2021. [eprint: 2101.06204](#).
- [TJS16] Jason Teutsch, Sanjay Jain, and Prateek Saxena. When cryptocurrencies mine their own business. In *Financial Cryptography and Data Security (FC 2016)*, February 2016.
- [TYE21] Itay Tsabary, Matan Yechieli, and Ittay Eyal. MAD-HTLC: Because HTLC is Crazy-Cheap to Attack. In *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part I*, pages 33–52. Springer, 2021.
- [USJ⁺18] Johanna Ullrich, Nicholas Stifter, **Aljosha Judmayer**, Adrian Dabrowski, and Edgar Weippl. Proof-of-Blackouts? How Proof-of-Work Cryptocurrencies Could Affect Power Grids. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 184–203. Springer, 2018.
- [VTL17] Yaron Velner, Jason Teutsch, and Loi Luu. Smart contracts make Bitcoin mining pools vulnerable. In *International Conference on Financial Cryptography and Data Security*, pages 298–316. Springer, 2017.
- [Vuk15] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *International Workshop on Open Problems in Network Security*, pages 112–125. Springer, 2015.

- [WHF19] Fredrik Winzer, Benjamin Herd, and Sebastian Faust. *Temporary Censorship Attacks in the Presence of Rational Miners*. 2019. Published: Cryptology ePrint Archive, Report 2019/748.
- [Wik] Bitcoin.it Wiki. Replace by fee in bitcoin. https://en.bitcoin.it/wiki/Replace_by_fee. Accessed 2020-12-23.
- [YSZ22] Aviv Yaish, Gilad Stern, and Aviv Zohar. Uncle Maker: (Time)Stamping Out The Competition in Ethereum. CESC '22, 2022.
- [Zca] Zcash. Homepage. <https://z.cash/>. Accessed: 2017-01-09.
- [ZHL⁺19] Alexei Zamyatin, Dominik Harz, Joshua Lind, Panayiotis Panayiotou, Arthur Gervais, and William Knottenbelt. Xclaim: Trustless, interoperable, cryptocurrency-backed assets. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 193–210. IEEE, 2019.
- [ZQC⁺21] Liyi Zhou, Kaihua Qin, Antoine Cully, Benjamin Livshits, and Arthur Gervais. On the Just-In-Time Discovery of Profit-Generating Transactions in DeFi Protocols. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 919–936, 2021.
- [ZQG21] Liyi Zhou, Kaihua Qin, and Arthur Gervais. A2MM: Mitigating Frontrunning, Transaction Reordering and Consensus Instability in Decentralized Exchanges. *arXiv:2106.07371 [cs]*, June 2021. arXiv: 2106.07371.
- [ZQT⁺20] Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc V. Le, and Arthur Gervais. High-Frequency Trading on Decentralized On-Chain Exchanges. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 428–445. IEEE, 2020.
- [ZSJ⁺18] Alexei Zamyatin, Nicholas Stifter, **Aljosha Judmayer**, Philipp Schindler, Edgar Weippl, and William J. Knottebelt. A Wild Velvet Fork Appears! Inclusive Blockchain Protocol Changes in Practice (Short). In *5th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security 18 (FC)*. Springer, 2018. (Short Paper).