Modelling the Dynamics of the Bitcoin Blockchain

by

Mabvuto Mwale

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science (Computer Science) in the Faculty of Science at Stellenbosch University

Supervisor: Prof. A.E Krzesinski

March 2016

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2016

Abstract

Modelling the Dynamics of the Bitcoin Blockchain

M. Mwale

Department of Mathematical Sciences,Computer Science Division,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MSc (Computer Science)

December 2015

Bitcoin is a peer to peer (P2P) electronic payment system proposed by Nakamoto in 2008. Central to the operation of Bitcoin is the blockchain, which is, in essence, a public ledger of all transactions. The blockchain is maintained by a group of volunteers called miners, who are rewarded with bitcoins for successfully mining blocks. Mining a block involves collecting and validating transactions, adding validated transactions to a block, and finally adding the block to the blockchain and broadcasting the block to the peer network.

The purpose of this thesis is to examine the profitability of block discarding attacks in the Bitcoin network. To this end, we developed a discrete event simulator to model the dynamics of the blockchain. We use a simplistic network model where all participants are miners. Initially, our investigation focusses on a model where all miners are honest, following the Bitcoin rules, and all block transmissions are subject to delays. We show that the delays cause forks to occur in the blockchain, which are quickly resolved. The simulation results for our network model closely agree with those observed from the real Bitcoin network.

We then examine a block discarding attack referred to as selfish-mine [32], where it is claimed that a small group of colluding miners can subvert the Bitcoin protocol. We evaluate this claim using relative revenue (the fraction of the total revenue that is credited to the pool of colluding miners) and confirmed blocks (the absolute number of blocks credited to the pool of colluding miners at the end of a mining period) for the case where there is instantaneous block transmission, and the case where block transmission is subject to delays. We show that this claim is true for the first case. However, for the latter case, selfish-mine is only profitable (in terms of the relative revenue) when the pool commands more than 30% of the total computing power of the network. We further show that a higher relative revenue does not necessarily entail a larger number of confirmed blocks and that the presence of selfish-mine causes both the honest and the dishonest miners to fare worse.

Finally we present a generalized form of selfish-mine, representing a greater variety of block discarding attacks, and use a genetic algorithm to search for an optimal configuration of the generalized selfish-mine that yields a better performance for the pool. We attempt to maximize either the relative revenue or the number of confirmed blocks. Our results show that the generalized form of selfish-mine when optimally configured yields better performance than standard selfish-mine, both in terms of the relative revenue and confirmed blocks.

Uittreksel

Modellering die dinamika van die Bitcoin Blockchain

("Modeling the Dynamics of the Bitcoin Blockchain")

M. Mwale
Departement Wiskundige Wetenskappe,Rekenaarwetenskap Afdeling,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MSc (Computer Science)

Desember 2015

Bitcoin is 'n stelsel vir elektroniese betalings van portuur tot portuur (P2P) waarmee Nakamoto in 2008 vorendag gekom het. Die werking van Bitcoin draai om die blokketting, wat in wese 'n openbare grootboek van alle transaksies is. Die blokketting word bygehou deur 'n groep vrywilligers wat as myners bekend is, wat met bitcoins beloon word indien hulle blokke suksesvol ontgin. Die ontginning van 'n blok behels die insameling en stawing van transaksies, waarna gestaafde transaksies by 'n blok bygevoeg, die blok aan die blokketting gehaak en dan na die P2Pnetwerk versend word.

Die doel van hierdie tesis is om die winsgewendheid van blokverwerpingsaanvalle in die Bitcoin-netwerk te ondersoek. Hiervoor is 'n diskrete gebeurtenissimuleerder ontwikkel om die dinamiek van die blokketting te modelleer. 'n Simplistiese netwerkmodel word gebruik waarin alle deelnemers myners is. Aanvanklik konsentreer die ondersoek op 'n model waarin alle myners eerlik is en die Bitcoin-reëls volg, en waarin alle blokversendings aan vertragings onderworpe is. Daar word aangetoon dat die vertragings vertakkings in die blokketting veroorsaak, wat vinnig uitgestryk word. Die simulasieresultate uit die netwerkmodel stem grootliks ooreen met wat in die werklike Bitcoinnetwerk waargeneem word.

Daarna word 'n blokverwerpingsaanval wat as 'selfsugtige ontginning' bekend is [32], ondersoek, waar 'n klein groep myners na bewering kan saamspan om die Bitcoin-protokol te ondermyn. Hierdie bewering word beoordeel aan die hand van relatiewe inkomste (die fraksie van totale inkomste wat aan die betrokke groep myners toegeskryf kan word) en bevestigde blokke (die absolute getal blokke wat aan die einde van 'n ontginningstydperk aan die groep myners toegeskryf kan word) in 'n scenario van onmiddellike blokversending sowel as ’n scenario waar blokversending aan vertragings onderworpe is. Daar word aangetoon dat hierdie bewering waar is in die eerste scenario. In die tweede scenario is selfsugtige ontginning slegs winsgewend (wat relatiewe inkomste betref) indien die groep meer as 30% van die algehele berekeningsvermoë van die netwerk beheer. Die studie toon ook dat ’n hoër relatiewe inkomste nie noodwendig ’n groter getal bevestigde blokke behels nie, en dat die aanwesigheid van selfsugtige ontginning eerlike sowel as oneerlike myners se prestasie verswak.

Laastens word ’n veralgemeende vorm van selfsugtige ontginning aangebied wat ’n groter verskeidenheid blokverwerpingsaanvalle voorstel. ’n Genetiese algoritme word gebruik om ’n optimale konfigurasie van die veralgemeende selfsugtige ontginning te bepaal wat ’n beter prestasie vir die groep oplewer. Hiervoor probeer die navorser hetsy die relatiewe inkomste of die getal bevestigde blokke maksimaliseer. Die resultate toon dat die veralgemeende vorm van selfsugtige ontginning in die optimale konfigurasie beter prestasie as standaardselfsugtige ontginning oplewer wat relatiewe inkomste sowel as bevestigde blokke betref.

Acknowledgements

I would like to express my sincere gratitude to the following people and organizations. Foremost, I would like to express my gratitude to my advisor, Prof Anthony E. Krzesinski, for his guidance on this journey. His attention to detail, critical nature, and input have been invaluable in making this work possible. In addition, I wish to thank Telkom for its support through the Telkom Centre of Excellence in Broadband Networks and their Applications, without which this work would not have been possible. Finally, I wish to express my gratitude to my family and friends for their support and encouragement throughout my studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Bitcoin[1] is a decentralized peer to peer electronic payment system based on cryptography or more accurately, a crypto-currency. The Bitcoin economy has grown at an incredibly fast rate with a current estimated market capitalization of about 3.5 billion US dollars since its introduction in 2009 [10]. While Bitcoin is still in its infancy and more of an experimental than an accepted currency, the trends suggest that it has the potential to shape the future of electronic payments.

In the Bitcoin system (to be discussed in detail later), mining is the activity which creates new bitcoins which are later put into circulation. Nodes called miners expend resources such as electricity on solving cryptographic puzzles and validate the Bitcoin transactions of other people. Verified transactions are grouped into what is known as a block. These blocks are added to the blockchain which is essentially a public distributed ledger of all Bitcoin transactions. It cannot be overemphasized that the integrity of the blockchain must be preserved at all times. The blockchain is maintained through consensus and the entity or entities with 51% of the total computing power of the network determine the correct version of the blockchain. The basic claim of Bitcoin is that as long as no one entity controls more than 50% of the total computing power of the network, the currency will remain decentralized [54]. It has been shown, however, that this claim relies on the premise that miners are honest and follow the protocol rules [4, 23, 32, 40, 65]. More recently, and compounded by the emergence of mining pools such as GHash.IO[2] which contributes close to 30% of the total computing power of the network at the time of writing,

[1]Throughout this thesis, Bitcoin (upper case) refers to the Bitcoin system while bitcoin (lower case) refers to the currency coin.

[2]GHash.IO is the largest Bitcoin mining pool which allows mining bitcoins with personal hardware or cloud-based mining power, collectively mining bitcoins worth a total exceeding $200 million in its first year of operation.

much attention has been paid to attacks on the Bitcoin network that can be broadly classified as block withholding and block discarding attacks [4].

1.2 Related Work

The idea of a successful subversive strategy with less than 50% of the total computing power of the Bitcoin network is not a recent one and can be traced to the early days of the system. One of the first attacks discussed is the double spending attack that Nakamoto [54] addressed in the original Bitcoin paper. Several subversive strategies have since been proposed and discussed over the years of Bitcoin's unprecedented growth compared to other crypto-currencies.

More recently, much attention has gone into strategies that are broadly classified as block withholding and block discarding attacks [4]. It is interesting to note that the same subversive strategy that [32] proposes is also discussed independently by [4]. While [32] calls it selfish-mine, [4] refers to it as st_1, a part of a larger st_k, $k = 0, 1, 2, \ldots, \infty$, family of block discarding attacks. The conclusion made by [4] is that the st_1 attack is profitable if the total computing power p of the attacker is

$$p > \frac{1 - ns}{3 - 2ns} \tag{1.1}$$

where ns is what they refer to as network superiority[3] and [32] concludes that selfish-mine is profitable when the total computing power α of the selfish pool is bounded as

$$\frac{1-\gamma}{3-2\gamma} < \alpha < \frac{1}{2} \tag{1.2}$$

where γ is the fraction of the honest nodes that mine on the published block (a previously secret block mined by the dishonest pool) when there is a race. A race[4] occurs when more than one competing block is added to the blockchain (one block having been mined by the attackers, the other block having been mined by an honest node) causing the blockchain to fork (have multiple branches of equal length) but eventually one branch wins (the branch on which the next block is mined). Notwithstanding the differences in approach and terminology, both analyses come to the same conclusion. It is further shown in [4] that any strategy st_{k+1} is more profitable than st_k. Thus st_2 or any st_k where $k > 1$, is more profitable than st_1 (selfish-mine), which in turn, is more profitable than st_0 (honest mining).

An analysis of the selfish-mine strategy is given in [23, 35]. The authors of [23] suggest that the claims of [32] are exaggerated, giving a number of reasons

[3]Defined as follows: when an honest miner mines a new block and the attacker is quickly informed of it and tries to release a competitive block as fast as she can, ns is the probability of the event that the next block mined by the honest network on top of either of the competitive blocks, will be on top of the attacker's block [23].

[4]It is a race for which branch gets extended in the next round of mining.

for their position. The key issues raised about the selfish-mine strategy as presented in [32] are:

1. The assumption that there is only one dishonest pool and that all members of the pool follow the pool rules. There is no incentive to prevent multiple pools from forming and we show later that this has a negative effect on the success of the selfish-mine strategy. In addition, just because members joined the pool does not mean that they will behave according to the pool rules.

2. There is no evidence that miners in the Bitcoin network as at present actually mine on the first block they received as claimed by [32] which is a key aspect of the proposed selfish-mine strategy. While Bitcoin is an open source project, currently the code of a great majority of miners is closed and to assume that no improvements or optimizations have been made would be presumptuous.

The remarks above are made by the authors of [23] without any supporting experimental results. On the other hand, the authors of [35] focus on both mathematical models and simulation models to analyze the performance of the selfish-mine strategy specifically under block propagation delays. They use a simple Markov model to track the states of the blockchain in the presence of block discarding attacks such as selfish-mine. In addition, the authors of [35] use a spatial Poisson model to analyze the values of γ in [32] and complement their analysis with simulation studies.

All the works mentioned above argue that the theoretical limit of the attackers' fraction of total computing power essential for the security of the system in the current Bitcoin specification is not 50% as claimed by Nakamoto, but much less. Bahack [4] claims that it is a little less than 25% and outlines proposals for protocol changes that can raise this limit to be as close to 50% as possible. On the other hand, [32] claims the threshold is as low as 10% and proposes changes that raise this threshold to 25%. Heilman [40] discusses the selfish-mine strategy and claims to raise the threshold of mining power necessary to profitably selfishly mine from 25% to 32% by employing the use of unforgeable timestamps. The author further claims that the changes proposed are robust even when the timestamps are forged.

The selfish-mine and other subversive strategies discussed above depend on the longest chain rule which is an inherent part of Bitcoin and most other Bitcoin-like systems. The longest chain is adopted by miners when there is a fork in the blockchain. A fork can be caused by block propagation delays or by an attacker following a subversive strategy. Decker and Wattenhofer [26] give a detailed analysis of block propagation delays and fork rate in the Bitcoin network. Since the honest miners cannot distinguish between an innocent fork and one caused by an attacker, the attacker who can extend his chain longer than the rest of the network can benefit from a subversive strategy. According to [22], this rule could, in fact, be an engineering mistake that other similar currencies have copied without deeper consideration of its implications.

Further work on attacks in the Bitcoin network can be found in [32, 43, 57, 63]. The authors of [57] and [63] perform an analysis of more optimal mining strategies than selfish-mine. In [63], the optimization of selfish-mine using an MDP approach is performed and it is further shown that adhering to the longest chain rule is not always the best option. The authors of [57] combine selfish-mine-like strategies with network level attacks such as an eclipse attack[5]. Both the authors of [57] and [63] conclude that there are more profitable strategies than selfish-mine as proposed in [32]. On the other hand, both [31] and [43] focus on attacks that are tailored towards pools (selfish or non selfish pools) in the Bitcoin network. While both [31] and [43] use game theoretical analysis, [43] focusses on DDoS attacks, and [31] on sabotage attacks that aim to reduce the revenue of the attacked pool.

1.3 Subject of Thesis

1.3.1 Background and Problem Statement

In 2013, Eyal and Sirer published a paper [32] in which a claim is made that Bitcoin mining is vulnerable and that a small group of colluding miners can subvert the protocol. The strategy presented in the paper is referred to as selfish-mine. Selfish-mine[6] if effective, allows a pool of colluding miners to earn more than their fair share of the mining revenue. The basic idea is that miners in the pool gain advantage by withholding blocks they discover from the rest of the network (honest miners) thereby causing the honest miners to waste their effort mining blocks that are destined not to be in the main branch of the blockchain. It is further claimed that a small group of miners commanding less than 10% of the total computing power of the network can subvert the protocol contrary to Bitcoin's claim that as long as the majority of miners remain honest, the protocol cannot be subverted. Bahack [4] discusses the same strategy although the threshold required for success is 25% as opposed to 10% of the total computing power. The model presented in [32] makes a number of assumptions. We make note of the following:

1. The model presented does not take block propagation delays into account. It is shown by Decker and Wattenhofer [26] that the average block propagation delay in the Bitcoin network is 12.6 seconds.

[5]An eclipse attack is an attack in P2P overlay networks where an attacker monopolizes all the incoming and outgoing connections of a victim thereby isolating the victim from the rest of the network.

[6]The terms selfish-mine, dishonest mining, and selfish mining are used interchangeably through this thesis.

2. It is assumed that only one pool in the network follows the selfish-mine strategy. Currently, there are several mining pools present in the Bitcoin network and as mentioned earlier, there is no reason to expect that only one pool would follow a block withholding strategy.

With the above assumptions and observations made in the section on related work, we are not convinced that the selfish-mine attack is as viable as presented in [32] let alone if it would be profitable in the real Bitcoin network. The purpose of this thesis is to examine these claims and verify whether the claims are accurate while using a model that more faithfully represents how the Bitcoin network operates. To that end, we wish to:

1. Investigate whether selfish-mine, as proposed in [32], is profitable when there are block propagation delays.

2. Verify the threshold required for selfish-mine to be profitable.

3. Investigate how profitable selfish-mine is when more than one pool is following the selfish-mine strategy and what effect this has on the threshold in 2 above.

4. Explore whether the selfish-mine strategy can be optimized to be more profitable.

Furthermore, we hope that the insight gained through this study will help in formulating means as to how these attacks can be detected and/or deterred.

1.3.2 Rationale

Bitcoin is not the only electronic currency in existence and neither the first to be conceived. However, the success of Bitcoin speaks for itself [6] and it has enormous potential to change how businesses conduct their business. Its application in international money transfers, pay as you go services, micropayments, dispute mediation, multi-signature accounts, online gaming, and online shopping is still being explored. Bitcoin is quickly gaining recognition on the African content. There are innovations such as BitPesa [12] in Kenya, Kipochi [20] in South Africa, BitPay [11] in many countries, including African countries such as Sierra Leone that demonstrate that Bitcoin can improve digital payment systems. Developing economies, especially in Africa are suited for Bitcoin due to the generally unstable currencies and high charges levied on customers. In addition, a lack of means to do online transactions is generally prevalent in Africa. Bitcoin, therefore, presents an opportunity to revolutionize this area. However, before Bitcoin can be fully accepted, it must be shown that the protocol is robust and secure, among other things. This will increase the level of confidence in the transactions and the overall Bitcoin economy, potentially making Bitcoin a platform that can change how electronic commerce and digital payments are currently being done notwithstanding the political, policy and legal issues that are yet to be fully resolved. We hope that this thesis will contribute towards demonstrating the stability or instability of Bitcoin and thereby contributing to future improvements to the system.

1.4 Thesis Outline

This thesis is arranged into seven chapters as follows:

- Chapter 1: Introduction
- Chapter 2: Currencies and Payment Systems
- Chapter 3: Bitcoin
- Chapter 4: Simulation of the Bitcoin Protocol
- Chapter 5: Selfish-Mine
- Chapter 6: Genetic Optimization of Selfish-Mine
- Chapter 7: Summary and Conclusions

This chapter gives an overview of the problem being addressed. Chapter 2 presents an outline of the evolution of currencies with a section dedicated to digital currencies and digital payment systems. Chapter 3 discusses Bitcoin and gives some details of its operation. In Chapter 4, the simulation framework and model is introduced and the results of simulation experiments of the Bitcoin protocol when all miners follow the protocol rules are presented. Chapter 5 brings us to selfish-mine. Here a discussion of the selfish-mine strategy and its evaluation using the simulation framework is presented. In Chapter 6 we discuss optimization of the selfish-mine strategy using Genetic Algorithm optimization. Finally, Chapter 7 gives a summary and conclusion.

Chapter 2

Currencies and Payment Systems

2.1 Currency: A historical Perspective

The need for people to exchange goods and services has long existed even before money as we know it today was invented. Throughout time, people have devised ways to make trade or exchange of goods more convenient. The barter system is the most ancient and primitive form of trade. In a barter economy, no notion of money existed. Goods were exchanged directly. One needed to find an individual who had the goods you were looking for and at the same time was in need of the goods you had. It is easy to see how difficult the exchange of goods was in this economy. Naturally people sought a better way of exchange; one that was universal. This gave rise to commodity currencies.

2.1.1 Commodity currencies

The idea of money originated in this era albeit in a primitive form. In this economy, there was some universal equivalent ("used as money") which was normally another good. This good was used to measure the value of all other goods. In ancient Europe, for example, they used cattle as the universal equivalent from whence the Latin word **pecunia** (meaning money) finds its roots[1]. The worth of any good would be assessed in comparison to cattle [25]. For example, 5 goats = 1 cow or 10kg corn = 1 cow. In this sense, cattle played the role of money. Other societies used leather, oil, beer, to mention a few examples.

2.1.2 Coins

Despite the improvement achieved through commodity currencies, the goods used as currency in the commodity currency economy were still not convenient to carry around due to their size and weight. Over time, the use of metal coins

[1]The word pecunia is derived from the Latin word **pecus** which means cattle.

as a medium of exchange was developed. The value of the metal coin was based on the intrinsic value of the metal used to make the coin. For example, a gold coin would have a higher value than a silver coin and other coins made from metals of lesser value. From around 1000 B.C, artifacts similar to coins such as small knives had been used in China. However, the first real coins were made and used around 560 B.C in a country called Lydia, a part of what is presently known as Turkey [5]. The coins were stamped with the King's seal. Similar innovations were taking place around the same time in cities in India, China and regions surrounding the Aegean sea.

2.1.3 Goldsmiths and Convertible Token Money

The Goldsmiths first introduced a concept that had the semblance of a central clearing authority for money [25]. In this economy, the Goldsmiths would keep your gold in their vaults and in exchange give you a piece of paper that certified that they owed you a certain amount of gold. This was an assurance to everyone that the bearer (of the paper) had a certain amount of gold in holding and hence the paper could be used to claim the actual gold. Consequently, the piece of paper could be used to trade.

With the passage of time, people started to use cheaper metals or pieces of paper that could be exchanged not for their intrinsic value but for the value that was printed on them. In this economy, the token (either a piece of paper or a metallic coin) was not worth much in that it had no intrinsic value, but it was used as "money" because it was backed by commodity money (gold or silver) at a fixed exchange rate. The token money would be exchanged with the issuer of the currency; the Goldsmiths. This is how the gold standard came into being.

2.1.4 Paper Money

The Chinese were the first to use paper money [7, 25, 39]. The "flying cash" of the Tang (618-907) dynasty named so because of its tendency to be blown by the wind was used around 800 A.D. Metal coins were then referred to as cash and not the paper money. The paper money was issued by the government originally for transit purposes when moving large amounts of cash over long distances since coins were heavy and inconvenient to carry. It was not a 'legal tender' but over time, merchants began using them as a convenient method of exchange [28].

The paper money was used during the S'ung (960-1279) dynasty and was extended to include paper certificates. Each note was sealed with a seal specific to one of the sixteen note-issuing houses making counterfeiting difficult. Using a mix of inks, the notes had pictures of houses, trees and people printed on them. They became widely circulated and eventually people started accepting them for many forms of payments. European explorers introduced the concept of paper money in Europe in the 13th century and John Law was a pioneer of paper money in Europe in the 1700s.

John Law [1] was an economist of Scottish origin, born into a family of bankers and goldsmiths. He was an avid gambler and originated economic ideas such as “the scarcity theory of value” and the “real bills doctrine”. Law is known to have espoused the following views,

- that money creation would stimulate the economy,
- that paper money is preferable to coins (which should be banned) and
- that shares are a superior form of money since they pay dividends.

In 1716, Law’s plan of a centralized bank that he presented to the Banque Générale in France was approved. This resulted in the founding of a private bank, Banque Générale Privé, that allowed investors to supply 25% of an investment in currency and the remainder in defunct government bonds. It was this bank which developed the use of paper money in France. In addition, the bank was allowed to issue its own currency backed by Louis of gold.

With this establishment and France’s investment in terms of company stock in the Mississippi Company, Law was able to make a reality of the monopoly companies he envisioned. However, the Mississippi company was a fraud that eventually failed and led to France being bankrupt and many individuals losing millions of livres. John Law died a bankrupt man. Around the same time, a similar public-private partnership undertaking was done by Great Britain that allowed the South Sea Company to fraudulently monopolize trade with South America. This venture met with an equally catastrophic failure resulting in the national economy of Great Britain being reduced considerably.

2.1.5 The Gold Standard

As was mentioned in previous sections, through time, various commodities such as gold and silver were used as money. Gold was used as money for thousands of years since it loses less value over time. This is because gold is generally a scarce commodity with a limited supply. The *gold standard* is one such monetary system in which the value of a unit of the currency is based on a fixed quantity of gold. Up until the 20^{th} century, one could exchange an amount of money for a fixed quantity of gold at central banks in most countries. The US abandoned the gold standard in 1971. The gold standard has since been abandoned by countries across the globe. In its place, we have fiat currencies.

2.1.6 Fiat Currencies

Fiat[2] money is a currency whose value is drawn from government regulation. Fiat money is distinct from other types of money, such as commodity money and representative money. Commodity money as the name suggests is based on a good (commodity). Usually, the commodity in question has other uses besides being a medium of exchange. Examples of commodities that have been used are gold and silver. Representative money, on the other hand, is a claim on the commodity rather than the actual good, for example, gold certificates [48]. Fiat money exists solely because of the government's power. Fiat money was introduced in China around 1000 AD. It has since been used by various countries, concurrently with commodity currencies.

2.2 Electronic Payment Systems

The advances in currencies or money have brought about the need to secure money and transactions through payment systems. A payment system [67] is an operational network that provides the means for exchange of monetary value between entities discharging mutual obligations, normally using bank deposits. It consists of the infrastructure and regulatory framework established to enable such mutual exchange to take place. The infrastructure consists of institutions and technical means [15] while the regulatory framework consists of the rules, procedures and standards. Payment systems include both electronic payment systems such as EFT or debit cards and the more traditional ones employ the use of physical instruments such as cheques.

"Traditional" payment systems could not cope with the needs of modern society, hence the emergence of their electronic counterparts. The growth of e-commerce resulting from the almost ubiquitous Internet has brought about the need for secure payment systems for transactions done on the Internet. Notwithstanding this, electronic payment systems go beyond a means for making payments on the Internet. Several kinds of electronic payment systems exist today. The important ones are briefly described below. A comparative overview of some of these payment systems is explored in [47].

2.2.1 Cards

Cards outweigh by far other forms of electronic payment systems in terms of usage and popularity. The types of cards found on the current market are credit, debit, and prepaid cards. Typically cards have a magnetic strip on the back where information about the card holder is recorded. More recently, cards are fitted with a smart chip. The card is normally issued by a financial

[2]Fiat is the Latin word for the third-person singular present active subjunctive of fiõ ("I become", "I am made").

institution which employs a central clearing authority for all transactions made using the card. With the services offered by Visa, MasterCard and Euro, one no longer needs to use a machine that is specific to the issuer (financial institution) making access to electronic money convenient. The card holder can use the card to draw cash or make payments at payment points while shopping for goods and/or services. Credit cards are particularly popular in more advanced economies because they typically allow payments which may not be available on some prepaid and debit cards.

2.2.2 Internet

A discussion on electronic payment systems cannot be complete without the mention of Internet payments. Many businesses have an online presence which makes shopping at a click of a button possible. Internet payments usually involve a person making a purchase online. When using Internet payments, customers can transfer money directly from their bank accounts (EFT) or use a card. Cards are usually preferred when making online purchases. As e-commerce grows, so will this payment system. Besides making purchases, Internet payments also include facilities that Internet banking normally offers such as SWIFT transfer [68].

2.2.3 Mobile Payments

Mobile banking is now almost a de facto standard for any bank. There is a constraint on the kind of transactions that can be carried out via a mobile phone. However, mobile phones still can be used to facilitate some electronic transactions. The customer can access their electronic money using a mobile phone, make payments, transfer funds, etc. Some mobile service providers have gone a step further to allow their customers to have a kind of a bank account based on the customer's MSISDN (Mobile Station International Subscriber Directory Number). Funds can be deposited into these accounts and then accessed later to carry out transactions normally through SMS/USSD[3]. This is generally referred to as mobile money. Examples are M-Pesa [73], MTN Money [53], Ecocash [29] to mention a few.

2.2.4 Person-to-Person Payments

Examples of institutions that facilitate such payments are PayPal [60] and Moneybookers [52]. When using these payment systems, the usual mechanism used as a store of value is a prepaid card. These mechanisms of the store of value are linked to an online account such as a PayPal account in such a way

[3]Unstructured Supplementary Services Data (USSD) is a protocol used in GSM networks, for example, when a user dials $*111\#$

that a person can pay another person using the online account as if the money were stored there. These provide an alternative for customers with no access to credit cards as these services can be accessed easily and securely over the Internet [16].

2.3 Electronic and Crypto-Currencies

A digital or electronic currency is an electronic medium of exchange. Most of the traditional money supply that is materialized as banknotes or coins has a digital side in that its value is often stored on a bank's computers. Due to our increasingly cashless society, one could argue that all currencies are becoming digital even though they may not be presented to us as such [30]. A digital currency exhibits properties similar to physical currencies, but may not necessarily exhibit all the properties, for example, its use may be accepted only in a restricted community. It allows for real-time transactions that are not constrained by physical borders [30]. Virtual currencies and crypto-currencies are types of electronic currencies.

1. A virtual currency is a non-governmental regulated digital currency that is usually accepted for use within a restricted community. It is controlled and issued by its developers. The US Department of Treasury in 2013 defined it more tersely as "a medium of exchange that operates like a currency in some environments but fails to meet all the attributes of real currency". It is clear from these definitions that a virtual currency lacks the key attribute of being legal tender (a legally recognized instrument for meeting financial obligations).

2. A crypto-currency is a medium of exchange that uses cryptography to secure transactions and to generate new currency units. It is a kind of alternative currency. Most crypto-currencies are decentralized peer to peer systems (as opposed to centralized electronic currency systems, such as PayPal) that often employ the use of a public ledger (such as the Bitcoin blockchain) to record transactions.

Bitcoin became the first decentralized crypto-currency in 2009. Numerous crypto-currencies have been created since then. The alternative crypto-currencies launched after the success of Bitcoin are frequently called altcoins. A few key ones are briefly discussed in the following sections and in Table 2.1 [33, 38].

2.3.1 Litecoin

The second largest crypto-currency in the world is Litecoin [38], created by Charlie Lee, an MIT graduate, and former Google engineer. It was launched 2 years after Bitcoin in 2011. Like Bitcoin, Litecoin [33] is based on an open

Table 2.1: Altcoins Summary.

Comparison of Bitcoin with Altcoins				
Currency	Deflationary	Blockchain	Proof of Work	Created
Bitcoin	Yes	Yes	SHA-256	2009
Primecoin	No	Yes	Cunningham chains	2009
Litecoin	Yes	Yes	Scrypt	2011
Peercoin	No	Yes	SHA-256	2012
Dogecoin	No	Yes	Scrypt	2013
Darkcoin	Yes	Yes	X11	2014

source global payment network that is not controlled by any central authority but uses “scrypt” as a proof of work. Unlike Bitcoin which requires specialized ASIC computers to demonstrate proof of work, CPUs of consumer grade can be used in Litecoin. It has a faster block generation rate as well as more rewards per block compared to Bitcoin [38].

2.3.2 Darkcoin

Launched in January 2014 [27], created and developed by Evan Duffield, Darkcoin is a more secretive version of Bitcoin. Although bitcoins are said to be anonymous, this is largely true only in comparison to traditional money. Since the blockchain keeps a history of all transactions ever carried out, these records can reveal a lot of information. On the other hand, Darkcoin works on a decentralized master-code network resulting in almost untraceable transactions hence offering more anonymity. It seems to be gaining popularity in a short span of time and Darkcoins, like Litecoins, can be mined (demonstration of proof of work) using a CPU or GPU [27].

2.3.3 Peercoin

Peercoin [46] which holds the third place in terms of cypto-currency market share [76], was developed by Sunny King (a pseudonym) and Scott Nadal and was launched in August 2012. It was the first digital currency to use a combination of proof-of-stake and proof-of-work. Peercoin is also referred to as PPCoin, Peer-to-Peer Coin, and P2P Coin. The initial generation of coins is performed with the usual hash based proof-of-work. The difficulty is expected to increase over time and this results in a change from using a proof-of-work to a proof of stake algorithm to generate new coins. The work required to generate blocks using the proof of stake algorithm is minimal [38]. As a consequence, the Peercoin network will consume less energy over time. Peercoin, unlike Bitcoin, is an inflationary currency since there is no fixed upper limit on the number of coins [33].

2.3.4 Dogecoin

Dogecoin [41] is another currency from the family of crypto-currencies that was launched in December 2013 and was created by Billy Markus and Jackson Palmer. Dogecoin is a modification of the Bitcoin protocol. Like Litecoin, it uses scrypt technology as a proof-of-work scheme. It has a block generation time of 60 seconds and the difficulty adjustment time is four hours. It is inflationary in nature similar to Peercoin so that the supply of coins will remain uncapped. Dogecoin deals with large numbers of lower value coins (individual coin value) consequently lowering the entry barrier to using the currency and making it a better fit for carrying out smaller transactions [33].

2.3.5 Primecoin

Primecoin [45] is by far the most distinct in the set of the crypto-currencies discussed in this section. Developed by Sunny King (who also developed Peercoin), its proof-of-work is based on prime numbers, making it distinct from Hashcash crypto-currencies based on the Bitcoin framework. This proof-of-work scheme is concerned with finding Cunningham chains and bi-twin chains. These are special long chains of prime numbers. Primecoin offers easier mining and greater security to the network [45].

2.4 Double-spending

Double-spending is when an entity manages to successfully spend the same money more than once. Where physical cash is concerned, once a dollar has been spent by its current owner, for example, it is no longer possible for that owner to spend the same dollar as the owner is no longer in possession of that dollar [42]. ‘Traditional’ electronic payment systems typically feature a centralized authority which records and validates all transactions. There are strict consistency rules applied by the central authority and transactions are recorded in the exact order in which they occurred. This makes a double-spending attack very difficult if not impossible to carry out. A similar consistency model exists in centralized virtual currencies such as WebMoney [42], thereby making them equally immune to double-spending attacks.

However, the same cannot be said of decentralized virtual currencies such as Bitcoin. By design, these distribute state (of the ledger) across a peer to peer network. Despite the advantage of increased privacy this brings about, the resulting consistency model is relatively weak consequently making the currency vulnerable to double-spending attacks. Nakamoto addresses this concern in the original Bitcoin paper [54] with a simple solution. The solution proposed is the consensus mechanism of the Bitcoin network. Once a transaction is confirmed, it is safe. For a transaction to be confirmed, it has to be 6 blocks deep in the blockchain. However, this can only work for transactions that are not in real-time. A window of opportunity to double spend still exists for fast payments. ETHZ researchers [44] performed an extensive analysis of this problem. More specifically, they show that not only can these attacks succeed with overwhelming probability, but also that, contrary to common belief, they do not incur any significant overhead on the attacker. They propose the propagation of double-spending alerts in the network and claim this would constitute a first important step towards efficiently detecting double-spending.

Chapter 3

Bitcoin

3.1 Introduction

Bitcoin is a P2P (peer to peer) electronic payment system. It is beyond the scope of this thesis to provide a comprehensive description of Bitcoin, its architecture and how it operates, but we endeavor in this chapter to outline the important aspects. For a more comprehensive treatment of the subject, refer to [2, 54, 58] and the Bitcoin wiki [49]. We begin by providing the key underlying technical details of Bitcoin (Section 3.1 to 3.3) and later discuss the overall Bitcoin ecosystem (Section 3.4).

3.1.1 Cryptography

Cryptography is the underlying technology behind Bitcoin hence its classification as a crypto-currency. Bitcoin employs cryptography to secure both transactions and user identities. Suppose Alice wants to send one bitcoin[1] to Bob. How does Bob know the coin belongs to Alice or that it is in fact Alice that sent the coin (spoofing)? How does Bob verify that Alice has not spent the said coin already (double-spending)? In a centralized currency system, the central clearing authority can keep track of coin ownership and verify transactions and user identities. However, since Bitcoin is a P2P system and has no concept of a central clearing agent, Bitcoin employs cryptographic hashes and digital signatures to address these issues.

3.1.2 The Bitcoin Wallet, Digital Signatures, and Ownership

Bitcoin uses public-key cryptography. Two cryptographic keys, one public and one private, are generated and stored in the wallet. At its most basic, a wallet is a collection of these keys thereby allowing a user to transact bitcoins. The

[1]Lower case bitcoin refers to the actual currency coin and not the entire system.

private key is used to sign transactions. Only the owner of the wallet has access to the private key. Consequently, this provides proof that the transaction signed with the said private key has been initiated by the owner of the wallet. It also makes the transaction tamper-proof once it has been issued. Getting back to our earlier example, public-key cryptography allows Bob to have proof that it is in fact Alice that sent the bitcoin and also that the transaction has not been altered by anybody since it was issued by Alice.

Wallets come in different forms depending on the type of device in use [2]. There are software wallets and physical wallets. Examples of physical wallets are paper wallets and hardware wallets such as trezor [71] and Ledger Nano [55]. These keep credentials offline while allowing a user to transact [2]. Software wallets are more popular as they are convenient to use. There are three kinds of software wallets, desktop wallets, mobile wallets and online wallets. Common implementations of online wallets are Snapcard [66], Circle [18], Coinbase [19] and Blockchain.info [14]. Desktop wallets are more numerous such as Bitcoin Core (full node and wallet software) [8] and mSIGNA [17] which have versions for Windows, Mac, and Linux. Mobile versions include Copay (available on Android, iOS, and Windows) [21], and Bitcoin Wallet (available on Android and Blackberry OS) [9].

3.2 The Blockchain

Wallets are often described as a place to hold or store bitcoins. However, due to the nature of the Bitcoin system, bitcoins cannot be separated from the blockchain. The blockchain is a data structure that contains an ordered list of blocks (Section 3.2.2). It is, in essence, the public distributed ledger of the Bitcoin system. Usually, the blockchain is stored as a flat file or in a simple database. Blocks in the blockchain are linked in such a way that each block refers to the previous block in the chain as shown in Figure 3.2. This rule does not apply to the first block in the chain, the genesis block (Section 3.2.3).

3.2.1 Transactions

A transaction is a transfer of value between Bitcoin wallets or more accurately, Bitcoin addresses. There are two kinds of transactions in the Bitcoin system, namely coin base, and regular transactions. Regular transactions take place when the value is transferred between two Bitcoin addresses. Going back to the Alice and Bob example, if Alice sends one bitcoin to Bob, that transaction must have three pieces of information besides other details:

1. An input. This field records the Bitcoin address from which Alice received the bitcoins in the first place (she received them from Eve). Eve in turn, received her coins from Jack (Jack's address is the input to Eve's transaction in which Alice's address is the output) who earned them as

a reward for mining a block. This way, any coin can be traced back to the coin base transaction that generated it.

2. An amount. This field holds the number of bitcoins being transferred and in this case, one bitcoin that Alice is sending to Bob.

3. An output. This field holds the address of the recipient of the value in the transaction, in this case, Bob's Bitcoin address.

All Bitcoin transactions contain the above three pieces of information except for the coin base transaction which does not require an input. This is a special transaction that represents the creation or minting of new coins. Transactions are broadcast to the network and the confirmation process begins when the next block is mined (which will contain the verified transactions).

3.2.2 Blocks

A block is a container data structure that aggregates transactions to be included in the blockchain. The block is comprised of a header and body. The header is 80 bytes in size and contains meta-data while the body contains a list of one or more transactions. The body makes up the bulk of the block and is at least 250 bytes in size. The average block contains more than 500 transactions, making it on average 1500 times larger than the header [2].

3.2.2.1 The Block Header

The block meta-data contained in the block header can be divided into three parts. The first item is a reference to the hash of the previous block (parent). A block connects to the blockchain through its parent, so this data item serves as a link. The second item is a set that is comprised of the difficulty, time-stamp, and nonce. These items relate to Bitcoin mining as detailed in Section 3.3. The last part contains the Merkle tree root (Section 3.2.2.3), a data structure used to efficiently and uniquely summarize all the transactions in the block.

3.2.2.2 Block Identifiers

A block can be identified in two ways. The primary identifier is its cryptographic hash (a digital fingerprint) commonly referred to as the block hash. This is a 32-byte hash made by hashing the block header twice using the SHA256 algorithm. The block hash identifies a block uniquely and unambiguously and can be independently derived by any node by hashing the block header.

A block can also be identified by its position in the blockchain, called the block height or depth. The calculation of the height or depth of a block starts from the genesis block which is at depth 0. Each subsequent block added after the first block is one position "deeper" in the blockchain. A common analogy is boxes stacked on top of each other [2].

The block depth is not a unique identifier as compared to the block hash. A single block will always have a specific block depth, but the reverse is not true. The block depth does not always identify a single block since two or more blocks might have the same block depth when the blockchain forks (see Section 3.2.4).

3.2.2.3 Merkle Tree

As mentioned in Section 3.2.2.1, each block in the blockchain contains a summary of all the transactions (Merkle root) in that block summarized using a Merkle tree.

A Merkle tree [50] named after its creator Ralph Merkle is a data structure used for efficiently summarizing and verifying the integrity of large sets of data. It is also known as a binary hash tree. Merkle trees are binary trees containing cryptographic hashes as illustrated in Figure 3.1. In the illustration, the data

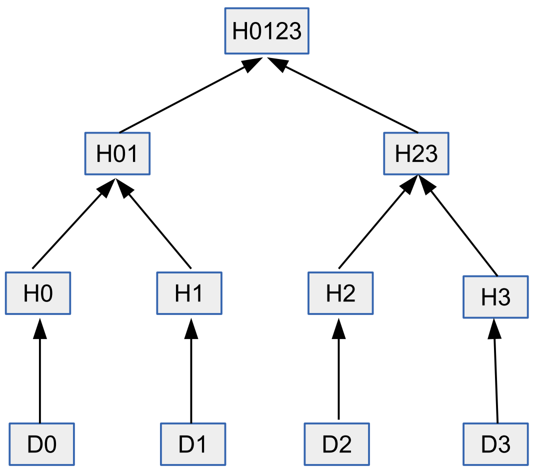


Figure 3.1: Example Merkle Hash.

items D0 to D3 are hashed individually, then the resulting hashes are hashed in pairs repeatedly until the top hash H0123 is computed which is referred to as the Merkle root. This is how Bitcoin hashes transactions until the Merkle root hash is computed and stored in the block header.

3.2.2.4 Linking Blocks in the Blockchain

Normally, full nodes (not devices like phones) on the Bitcoin network maintain a complete local copy of the blockchain[2]. As new blocks are mined, they are broadcast to the network. Once a node receives a new block, it updates the

[2]Starting at the genesis block.

local copy of the blockchain and extends it accordingly. However, before the block is added to the blockchain, it is validated. Validation involves verifying that the block is valid, that transactions included in the block are valid and that a valid proof of work exists (Section 3.3). Once validation is complete, the block is linked to its parent by examining the block header and looking for the previous block hash.

3.2.3 The Genesis Block

The genesis block is the first block in the blockchain and it was created in 2009. All blocks in the blockchain can be traced back to the genesis block. This block cannot be altered as it is encoded within the bitcoin client. This is the starting point for the blockchain, a secure "root" from which to build a trusted blockchain. As [2] puts it,

> *"Every node always "knows" the genesis block's hash and structure, the fixed time it was created, and even the single transaction within."*

This means that every node starts with at least a blockchain containing one block.

3.2.4 Blockchain Dynamics

Each block in the blockchain has a unique identifier, its hash (the hash of the its header) computed using the SHA256 algorithm. Each block also has a field referred to as the "previous block hash" within its header. This field serves as a pointer to the previous block, known as the parent block. Since the blockchain is a tree-like data structure, these pointers link blocks in such a way that a block points to its parent, which in turn points to its parent creating a chain going back to the root of the tree, the genesis block.

In the ideal scenario, the blockchain can be represented as a unary tree with a single branch extending from the genesis block. In this case, each block has exactly one parent and each parent has exactly one child. However, in the real Bitcoin network, forks in the blockchain occur occasionally, mostly due to communication delays in the network [26]. A fork arises when several blocks are mined almost simultaneously by different miners[3]. With or without the presence of forks, a block has just one parent. However, a parent may have multiple children when the blockchain forks. This is because the blocks mined almost simultaneously all point to one block as the parent. A fork is shown in the blockchain schematic in Figure 3.2. Forks are resolved using the longest chain rule. When the next block is mined, it will be mined on either of

[3]A miner is a node on the network that verifies transactions and is involved in the consensus mechanism of the Bitcoin system.

the competing blocks (B4-B or B4-A), making one branch of the fork longer as shown in Figure 3.2. According to the current Bitcoin specification, when one branch becomes one block longer than the other branch (in this case, the branch comprising block B4-B and B5), all miners abandon the shorter branch (B4-A) and begin mining on the longer branch.

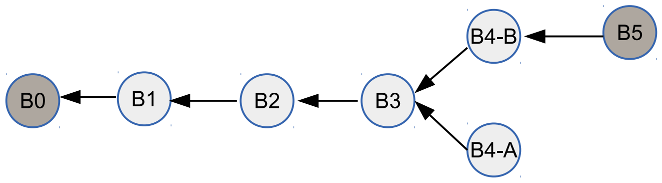


Figure 3.2: Visualization of the blockchain with a fork.

Since a block's hash is computed on its header which contains the hash of the parent block, it follows that a change in the parent's hash or identity affects the hash of the current block. When the parent is modified in any way, the parent's hash changes. The changed hash of the parent necessitates a change in the "previous block hash" field (pointer) of the child. This causes the child's hash to change, thereby necessitating a change in the grandchild's pointer. The resulting required change in grandchild, further necessitates a change in the great grandchild and so on. The cascading effect caused by this daisy chain linking of blocks ensures that once a block has many blocks following it, a change to the block makes the recalculation of all subsequent blocks necessary. The enormous computation that such a recalculation would require entails that a long chain of blocks makes the blockchain history for all practical purposes permanent. It is generally accepted that once a block is 6 blocks deep in the blockchain, then it is tentatively confirmed. While it remains true that the possibility of a block being reversed or the chain being undone by a longer chain exits, the probability of such an event happening due to the amount of computation that it would require decreases with time until it becomes negligible. This is a key feature of Bitcoin security.

3.3 Mining

Mining is the process by which transaction records are added to the blockchain. Bitcoin nodes use the blockchain to verify whether a coin is legitimate and if it has not been spent already.

It is not accidental that the mining process is resource-intensive. It was designed this way to regulate the block generation rate in the network. Each block that is mined is checked to see if it has a valid proof of work. This is the reason why it is called mining as it resembles the mining of commodities such as gold (the bitcoin miner exerts effort to compute the proof of work like a gold miner exerts effort to mine gold) and the new coins are analogous to the commodity being mined. In the initial phase of Bitcoin, a miner needed only a PC to solve the cryptographic puzzle that forms the proof of work. However, as time went on and more miners appeared, the difficulty of the cryptographic puzzle was increased so that the average time to mine a block remained at its designed value of 10 minutes and this was met by the introduction of Graphical Processing Units (GPUs) in the mining process. The difficulty was further increased and consequently miners migrated to using General Programmable Field Arrays (GPFAs) computers. This necessitated a further increase in the difficulty of the puzzle and in response, miners resorted to using Application Specific Integrated Circuit (ASIC) computers to mine.

Bitcoin follows a democratic voting process to decide what version of the blockchain is considered correct. Mining is the process by which, for all practical purposes, this "secure tamper-resistant" consensus is reached. To provide an incentive to miners to validate transactions and secure the Bitcoin system, a reward in the form of a fixed amount of bitcoins (currently 25 bitcoins) is given to the successful miner of a new block. This is how new coins are introduced to the system. Miners also collect transaction fees for each transaction they validate.

3.3.1 The Mining Process

A summary of the important features of the mining process is shown in the flowchart in Figure 3.3. The process involves collecting and validating transactions. The valid transactions are hashed and then the miner attempts to compute a proof of work. This process goes on until either a valid proof of work is found or the miner receives a new block mined by another node. At this point, the miner adds the new block to its blockchain and begins to mine on the new block.

3.3.2 Proof of Work

A proof of work (POW) is a system that requires a service requester to perform some minimum required work (usually some computation that requires processing time by a computer). It is often used to deter attacks and service abuses. The concept can be credited to Dwork and Naor [56]. There are several kinds of POW schemes, but a key feature of them all is their asymmetry. It is critical for any POW scheme that the work done by the requester must be moderately hard (costly or time-consuming but feasible) but trivial to check for the service provider. Bitcoin's proof of work is based on Back's Hashcash [3]. Its premise is the computation of a cryptographic puzzle that involves generating a hash of the block header to meet specified difficulty requirements. The computation of the crypto-puzzle is a random process with

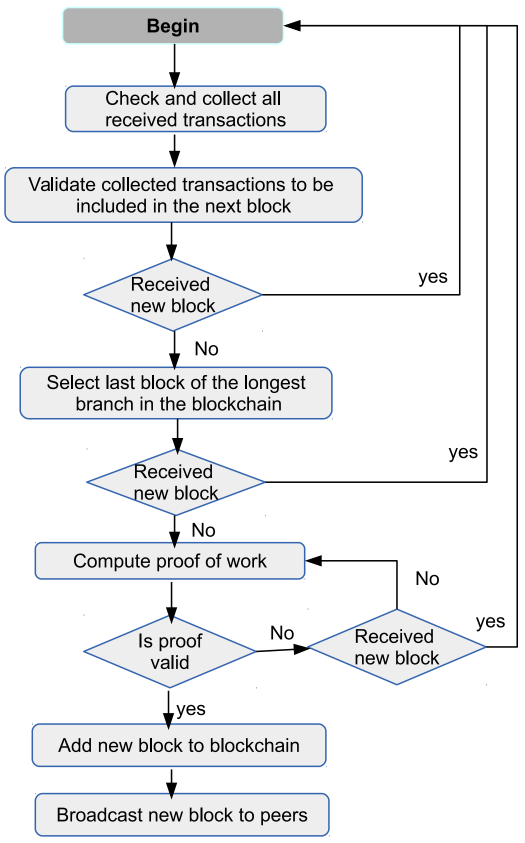

Begin
Check and collect all received transactions
Validate collected transactions to be included in the next block
Received new block
yes
No
Select last block of the longest branch in the blockchain
Received new block
yes
No
Compute proof of work
No
Is proof valid
No
Received new block
yes
yes
Add new block to blockchain
Broadcast new block to peers

Figure 3.3: Mining Process Flowchart.

a low probability of success. Consequently, to generate a valid POW requires a substantial amount of trial and error.

Consider a miner M. To compute the proof of work, M attempts to compute a double-SHA256 hash H of a block header such that the number X of leading zeros in H is

$$X \geq T \tag{3.1}$$

where T is the target difficulty (number of leading zeros). The value of T is adjusted from time to time to make the computation more difficult or easier to ensure that a block is mined every 10 minutes on average. H is computed as below

$$H = M(n_i, B_h, H_{i-1}) \tag{3.2}$$

where H_{i-1} is the double-SHA256 hash of the parent block, B_h is the Merkle root (Section 3.2.2.3) of all transactions in the current block and n_i is the nonce.

All three values B_h, n_i and H_{i-1} are concatenated to form a string that is used during the computation. The node computes H with nonce $n_i = 0$. If X does not have the required number of leading zeros, $n_i := n_i + 1$ and H is recomputed. This is repeated until X satisfies equation (3.1).

If all possible nonce values have been exhausted and a POW that meets the minimum difficulty requirement has not been found, changes are made to the coin-base field (coinbase) in the coin base transaction. This changes the current block header and in turn, the Merkle tree is recomputed which results in a different Merkle root B_h. In this way, the coin-base field serves as a second source of randomization for the POW computation [2].

3.4 The Bitcoin Ecosystem

Having laid the technical foundation for Bitcoin in the previous sections, we can now focus our attention on the Bitcoin ecosystem. The Bitcoin ecosystem goes beyond the technology that enables the distributed validation and processing of transactions. It is a complex Econo-Sociotechnical system [24] that is comprised of the core Bitcoin network and intermediary entities such as payment processors and wallet firms. An analysis of the Bitcoin ecosystem based on systems theory is given in [24].

Bitcoin has received a substantial amount of media attention, both positive and negative. However, the majority of the negative media reports have had more to do with intermediary systems built on top of Bitcoin than the underlying Bitcoin protocol itself. Intermediaries of note in the Bitcoin ecosystem are payment processors, digital currency exchanges, and wallet firms. These have been mostly successful. A good example is BitPay [11] which has relationships with more than 60,000 merchants since its founding in 2011 making it the largest Bitcoin payment processor. However, despite these successes, failures like that of Mt.Gox, the largest exchange and wallet service, have raised issues about the unregulated nature of the system besides the resulting negative media reports and perception. About 470 million US dollars worth of bitcoins at the time were lost from customer accounts held at Mt.Gox when it collapsed [69] in February 2014. Since this failure, it is clear that a paradox exists for the Bitcoin community since the original idea of Bitcoin was to avoid the introduction of regulation, but such incidents demand a closer regulatory framework to secure bitcoins and restore confidence in the overall Bitcoin economy.

Finally, the debate as to whether bitcoins should be considered currency or commodity continues and it remains unclear with which perspective the average consumer and the monetary authorities will eventually view bitcoins.

3.4.1 Acquiring Bitcoins

Like any new technology, Bitcoin is not yet widely accepted in the business community, but numerous merchants now accept bitcoins as payment for goods or services although the means and ways to use bitcoins to pay for services is still an active area of research and development. It may soon be possible to use bitcoins on PayPal [61] to make purchases. However, before one can use bitcoins, the bitcoins must be acquired. Bitcoins can be obtained by,

- buying them from an exchange,
- getting them from an individual who has bitcoins or
- receiving a reward in bitcoins for successfully mining a block.

In the case where an individual intends to buy bitcoins, there are several digital currency exchanges worldwide that trade in bitcoins. Some of the popular ones are Bitstamp, Bitfinex, Coinbase, Cryptsy, BTC-e, Kraken, and BTCChina. The majority of bitcoin exchange transactions is done in US dollars.

3.4.2 Scalability

Scalability is an important concern in any system. Considering the volumes of transactions processed by current electronic payment systems such as Visa (record of about 2000 TPS), can Bitcoin scale to accommodate such volumes of transactions? This is a critical concern for the following reasons.

- The size of the block is currently limited to 1MB. The average size of a Bitcoin transaction is 250 bytes. This means at most an average of 4194 transactions can be incorporated into each block. Since a block is mined every 600 seconds on average, it means with the current block size, Bitcoin transaction rate is capped at about $4194/600 \approx 7$ TPS.
- All full nodes keep a copy of the entire blockchain which is currently about 20GB in size and grows by one block every 10 minutes on average. Increasing the size of the block to accommodate more transaction results in an increase in the rate at which the size of the blockchain increases. Consequently, the power (storage[4] and processing) of full nodes will have to be increased. This raises the cost of having full nodes, which in turn, has a centralizing effect as less powerful nodes will eventually leave the network. Increasing the block size limit has long been an issue of contention in the Bitcoin community. Recently, as reported in the 22^{nd} August 2015 edition of The Economist [70], two of the main developers of Bitcoin have released a competing version of the Bitcoin core software that increases the block size limit. The new software is called Bitcoin XT and it is envisioned that once 75% of the miners migrate to Bitcoin XT, there will be an upgrade to increase the block size limit to 8MB, which will be doubled every two years.

[4]In order to allow for 2 transactions per day for every individual alive, each block would need to be about 24GB in size [49].

- Transactions in a block are only considered confirmed (tentatively) when the block is 6 blocks deep into the blockchain. This means a transaction will only be confirmed on average after an hour.

The above limitations warrant serious concern should Bitcoin be widely accepted for use. For example, if a customer is making a purchase online using bitcoins, do they have to wait for an hour before the purchase is confirmed? A number of solutions have been proposed for the above issues such using the Lightning Network, which allows trust-less off-chain transactions. For a thorough treatment of the subject, refer to the Bitcoin wiki [49].

3.4.3 A Comparison of Bitcoin and Other Payment Systems

Bitcoin is an electronic payment system [54]. An overview of electronic payment systems is given in Chapter 2. In this section, we compare Bitcoin to other payment systems, specifically Visa [72], Paypal [60] and M-Pesa [64]. We begin by briefly explaining how the other payment systems work.

3.4.3.1 Visa

There are 5 key entities that enable typical Visa [72] transactions to take place.

- *Cardholder*: A customer that has been issued with a Visa-branded card.
- *Merchant*: an entity that accepts Visa payments. A typical merchant would be a store such as a supermarket or a service provider such as an airline.
- *Issuer*: The issuer is the financial institution that issues the card to the cardholder.
- *Acquirer*: This is the financial institution that provides financial services to the merchant in question for purposes of processing the merchant's payments.
- *VisaNet*: The network that provides fast, reliable and secure connections for the financial institutions (Issuers and Acquirers) to exchange financial information.

The customer (card holder) initiates a transaction from a merchant site, an ATM (overseas ATM) or a pay point in a store. Details of the customer are captured from the card. If the customer uses an ATM or pay point, the details are read from the card's chip or magnetic strip and the customer's personal identification number (PIN) is requested for authentication. If a transaction is initiated online, the user enters the card details which include the 16 digit primary account number (PAN), expiry date and sometimes the 3 digit security code. Once the required customer details are captured, an authorization request is created. This is a message that contains the customer details, transaction details, and merchant identification number. The authorization request is then forwarded to the acquirer. The acquirer records the transaction and forwards the request to the appropriate network identified by the first few digits of the PAN, in this case, VisaNet. VisaNet records the request and forwards it to the issuer identified by the bank identification number (BIN) which is a part of the PAN. VisaNet can process more than 56000 TPS (transactions per second) [72], the average being 2000 TPS and a daily peak of about 4000 TPS. The issuer receives and records the request. The issuer must either decline or authorize the request. To achieve this, a number of checks are done, but at a minimum and depending on the type of card (credit, debit or prepaid), the issuer verifies that the cardholder has enough credit to cover the transaction. If so, the request is authorized and recorded in the ledger. The request is forwarded back through the same channel and each entity on the return path records the status. If the request was authorized, the transaction is finalized when the acquirer pays the merchant or pays the card holder in case it is a cash withdrawal otherwise an error is returned to the cardholder holder. This process completes normally within seconds. The issuer debits the customer's account and reimburses the acquirer the amount paid to the merchant on behalf of its customer. The merchant's account is credited after a delay of several days.

3.4.3.2 PayPal

PayPal [60] is a person to person payment system (Chapter 2 Section 2.2.4). PayPal allows registered users to pay for goods or services online from registered merchants and to send money. Cash transactions such as withdrawals are not supported. There are three kinds of PayPal accounts.

- *Personal*: A personal account is used by individuals who want to make online payments. There are no opening fees for this type of account.
- *Premier*: This is used by individuals who want to sell things online. It allows for the transfer of funds from the PayPal account into the individual's bank account.
- *Business or Merchant*: This is used by merchants such as e-bay.

The advantage of PayPal is that it charges a low fee for any transaction for all account types as compared to Visa. A PayPal account can be linked to either a bank account, debit/credit card or prepaid account. PayPal acts as the middle man to transfer payments between the buyer (personal account) and seller (merchant). Consequently, the back-end process when processing a payment request will differ depending on the mechanism, e.g. ACH (Automated Clearing House). The personal account holder normally initiates a payment request by selecting pay with PayPal on the merchant site. This directs the user to the PayPal site where user details are entered. Once user verification is done, PayPal performs a number of checks depending on the payment mechanism such as account status checks if using ACH. Once the necessary checks are done, PayPal generates a payment token that is forwarded to the merchant's site for approval. Upon merchant confirmation, PayPal records and transmits an appropriate transaction type, e.g. a debit order to the personal account holder if using ACH. The merchant's account is then credited with the transaction amount. This process, like Visa, is also completed in a matter of seconds. Transactions fees are often posted later to the merchant's account.

3.4.3.3 M-Pesa

M-Pesa[5] is a mobile payment system (Chapter 2 Section 2.2.3). It was launched in 2007 for Safaricom in Kenya and Vodacom in South Africa. It has since been implemented in other countries [64]. For any transaction to take place, there are normally three parties involved.

- *End User*: A customer with a mobile phone registered with the service provider. The end user has an account attached to the mobile phone number.
- *Mobile Services provider*: The mobile telecommunication company, in this case, Vodacom or any other provider that has deployed M-Pesa.
- *Merchant*: An entity that accepts mobile payments that is registered with the service provider.

Typically, M-Pesa allows an end user to perform transactions such as money transfers, cash deposits, cash withdraws, bill payments and to make purchases from a merchant using funds from the account attached to their mobile number. For this reason, the service provider needs a separate banking license (separate from the telecommunications license) with its accompanying regulations to operate such a service. The service can be accessed using USSD[6], a mobile phone application or STK (SIM Tool Kit) menu. It is from this interface that the end user initiates a transaction. However, money must be deposited into the account attached to the mobile number at any authorized agent before the 'mobile money' can be used. When the end user initiates a transaction from the mobile phone, the transaction is received and processed by the mobile service provider typically the same way transactions are processed at a

[5]M representing mobile and pesa is a Swahili word which means money

[6]Unstructured Supplementary Services Data (USSD) is a protocol used in GSM networks, for example, when a user dials ∗111#

traditional bank. When the service provider verifies that the user has enough credit in their account to offset the transaction amount, the transaction is authorized. The merchant's account (or a recipient end user's account in case of money transfer between end users) is credited with the transaction amount, and the initiating end user's account is debited with the transaction amount plus service fees. The merchant or recipient end user receives notification of payment and the initiating end user is informed of the transaction's status, success or failure. The process typically completes within seconds.

3.4.3.4 Bitcoin

Typically, a Bitcoin transaction involves three entities.

- *End User*: The customer who wants to pay for goods or services with bitcoins.
- *Bitcoin Wallet Software*: The software that stores the user's bitcoins and the private key used to sign transactions. Wallets are discussed in Section 3.1.2.
- *Bitcoin Core Network*: The core P2P network of miners maintaining the blockchain.
- *Merchant*: An entity that accepts bitcoins as payment for goods or services.

Since there are varying implementations of how payments using bitcoins are done such as BitPay [11], BitPesa [12] and PayFast [59], we use PayFast for illustrative purposes. An end user will initiate a transaction from a merchant site by clicking the "pay with bitcoin" option. This redirects the customer to the PayFast site which generates the QR code that the customer can scan with their mobile Bitcoin client. The QR code will have the amount of the purchase in bitcoins at the current exchange rate as provided by BitX [13] and the merchant wallet address. For other kinds of wallet implementations, the amount in bitcoins and the merchant's wallet address are also displayed on the PayFast site. The user then sends the funds to the merchant by signing the transaction with their private key and the transaction is broadcast to the Bitcoin core network. Eventually, PayFast receives confirmation from BitX that the funds have been received. The user then gets a success notification and is then redirected back to the merchant site. After this, Payfast will notify the merchant that the payment has been received. The transaction is then considered complete. Depending on the amount involved, some merchants may opt to wait until the transaction is confirmed by the Bitcoin core network. This entails that the transaction will not be finalized until after an hour. Table 3.1 gives a summary of some differences between Bitcoin, M-Pesa, PayPal, and Visa.

Table 3.1: A Comparison of Bitcoin and other Payment Systems.

System	Pay Points	Online	Cash Withdraw	Send Money	Cost
Visa	Yes	Yes	Yes	No	High
PayPal	No	Yes	No	Yes	Medium
M-Pesa	Yes	No	Yes	Yes	Low
Bitcoin	Possibly [a]	Yes	Possibly [a]	Yes	Low

[a] Possibly means there may be no current implementations of such a facility but it is theoretically possible.

Bitcoin works like any other payment system with the advantage of low cost. However, the biggest problem remains the scalability issues mentioned in Section 3.4.2.

Chapter 4

Simulation of the Bitcoin Protocol

4.1 The Simulator

Modelling and simulation are important and generally accepted as ways to represent and explore the behaviour of diverse physical and non-physical systems. In order to gain insight into the dynamics of the Bitcoin blockchain, we developed a discrete event simulator in C++. The simulator models the important aspects of the Bitcoin P2P overlay network, mining and block transmission. The rules described in the following section summarize the significant aspects of our simulation model but are by no means exhaustive. We assume at this stage that only honest miners who follow the prescribed Bitcoin rules are present in the network and that all communication is subject to communication delays.

4.1.1 The Simulation Model

Our simulation model implements a subset of the protocol rules as described in the protocol specification presented in the Bitcoin wiki [49]. Our model does not include generation and transmission of transactions as these details are not relevant to our investigation. Consequently, all the rules on transactions have been omitted and we assume that transactions are valid and verified before they are added to a block. A summary of the simplified set of rules implemented in the simulator is presented in Appendix A. Based on this, our Bitcoin model works as follows.

- Blocks are mined at the instants of a Poisson process. One block is mined every 10 minutes on average.
- A total of 10,000 blocks are mined. This includes the blocks that are deleted due to forks.
- All nodes have uniform hashing power and have the same probability of being chosen to mine a block.

- Nodes are located at random on an (x, y) plane
- The network communication delay t seconds is a random variable sampled from a normal distribution whose mean is proportional to the Euclidean distance between the two nodes communicating and with a constant coefficient of variation (CV). Unless otherwise stated, $CV = 0.1$ in all our experiments.
- When a node (miner) mines a block, the block is first added to its local blockchain, then it is broadcast to all the peers in the network subject to a communication delay t. In the real Bitcoin system, only the header is sent and a node requests the block data if it does not already have the block.
- When a node receives a block, there are two possible scenarios.
    1. The block has a valid solution and its parent is in the blockchain. The block is appended to its parent.
    2. The block has a valid solution, but its parent is not in the blockchain. The block is added to the block pool [1] and its parent requested.
- The block pool is checked to see if there are orphaned blocks whose parent pointer points to a block in the blockchain. Such blocks are removed from the block pool and added to the blockchain.
- A received block that is already in either the blockchain or block pool is considered a duplicate and deleted. Duplicate blocks are possible because a block can arrive at a node before its parent (which is still in transit). When this happens, the parent is requested causing the same block to be sent to one node more than once.
- A k-nary tree is used to represent the blockchain [62].

4.2 Simulation Results

Based on the model above, we simulated a network of 1000 nodes (miners). Multiple replications of the simulation were executed with different random number seeds. In our case, 12 replications were run for each simulation and 95% confidence intervals were computed. The results are presented in Figures 4.1 to 4.3.

[1] The block pool is the container that stores the blocks that temporarily cannot be appended to the miner's blockchain.

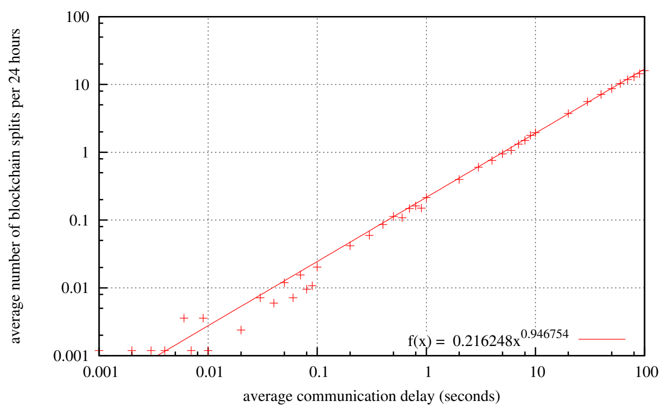


Figure 4.1: Blockchain splits per day as a function of the communication delay.

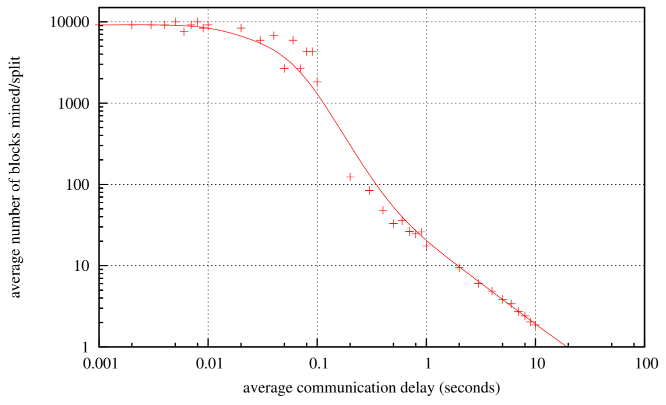


Figure 4.2: The ratio $r(t)$ as a function of the communication delay.

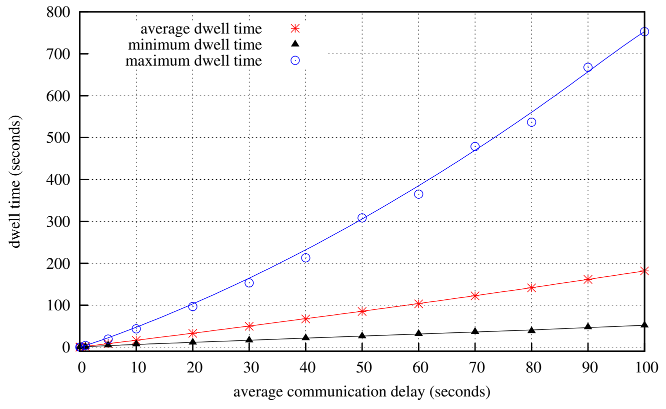


Figure 4.3: The dwell time t_d as a function of the communication delay.

Figure 4.1 shows a plot of the number of splits per day as a function of the communication delay in seconds. Both the x and y-axis are logarithmic.

Figure 4.2 shows the ratio $r(t)$ of the number of blocks mined to the number of splits as a function of the communication delay t.

Figure 4.3 shows a plot of the dwell time (t_d) as a function of the communication delay. The Bitcoin protocol works in such a way that the blockchains at each node in the network are usually identical, having a single leaf. However when communications delays are present, differences in the blockchains at nodes can be observed. These differences are resolved with time. Consider a time T when the blockchains at all the nodes are identical. Blocks are mined and added, further extending the blockchain. The blockchains may now differ at the nodes. Let T' such that $T' > T$ denote the next time when all blockchains are synchronized again. The dwell time t_d is defined as

$$t_d = T' - T. \tag{4.1}$$

4.3 Discussion

From Figure 4.1, we can see that the relationship between the number of splits and the communication delay is linear. When communication delays are absent, the blockchain never splits since blocks are received instantaneously the moment they are transmitted[2]. However, this deviates from reality since all real networks have latency. Decker and Wattenhofer [26] show that the average block propagation delay in the Bitcoin network is 12.6 seconds. Usually, splits occur when two blocks are mined at almost the same time. Due to block propagation delays, the nodes are not instantly aware of a block that may have been mined hence they continue to mine on top of the last block received. The larger the delay, the longer it takes for the entire network to be aware of any new block(s). Conversely, the smaller the delay, the quicker the blockchains at individual nodes get synchronized resulting in each node quickly having a correct and complete view of the blockchain. As a result, each node is almost always mining on the correct block of the blockchain (longest chain) resulting in fewer splits. This is the reason for the observed linear relationship between the number of splits and communication delay.

The dwell time is a measure of how quickly a block propagates through the network. From Figure 4.3, we can see that the average dwell time $t_d \sim kt$ where k is a constant and t is the average communication delay. We can also see that when $t \approx 12$, the maximum dwell time is about 45 seconds. This agrees with Decker and Wattenhofer's [26] findings that even after 40 seconds, some nodes in the Bitcoin network will not have received the new block.

[2]In our simulation model, blocks are mined at instants of a Poison process, making it impossible for more than one block to be mined at the same instant. However, in the real Bitcoin network, it is possible that more than one miner can mine a block at the same instant making splits possible even when there are no communication delays.

In addition, our results agree with Decker and Wattenhofer's [26], who observe a 1.69% fork rate in the observed 10,000 block interval with an average block propagation delay of 12.6 seconds. Our simulation results show that when 10,000 blocks are mined and the average communication delay is 10 seconds, an average of 2.4 splits per 24 hours is observed. Since an average of 144 blocks are mined per 24 hours, our percentage fork rate is calculated as, $(2.4/144) * 100 = 1.67\%$. This is close to Decker and Wattenhofer's value of 1.69% and confirms that for the network model under investigation, our simulator accurately models the Bitcoin blockchain.

We therefore conclude that, when all nodes in the network are honest and communication delays are present, the Bitcoin protocol is robust and can resolve the splits and differences that occur on the individual blockchains so that a consistent view of the blockchain is observed across the network.

Chapter 5

Selfish-Mine

5.1 Introduction

Selfish-mine is one of several strategies that have been developed to subvert the Bitcoin protocol [4]. According to [32], Bitcoin mining is not incentive-compatible[1] and this can lead to the emergence of mining pools that follow the selfish-mine strategy to gain advantage and earn revenue in excess of their fair share. Key to the operation of the selfish-mine strategy is the selective and opportunistic revelation of blocks discovered or mined by members of the pool. This causes the honest nodes to waste their resources mining blocks that are destined not to be in the main chain. The main chain is the branch of the blockchain consisting a chain of blocks (linked by pointers to the previous block) with the largest sum of work done in expectation, beginning at the genesis block.

To specify the selfish-mine strategy, we assume that the miners are divided into two groups [32], the honest nodes following the Bitcoin rules and a minority of dishonest nodes following the selfish-mine strategy[2]. We refer to the fraction of the total computing power commanded by the selfish pool as alpha (α). The detailed operation of the selfish-mine strategy is shown in Algorithm 1 and the state transition diagram in Figure 5.4.

Initially, the selfish pool has not discovered any blocks, so the dishonest nodes in the pool mine on the last public block received (block with the serial number n_p). The serial number of a block is synonymous with its position (depth) in the blockchain. This is shown in lines 2 to 5 of Algorithm 1 and

[1]In mechanism design, a process is incentive-compatible (IC) if the honest revelation by participants of any private information requested by the mechanism results in the participants faring best [74]. Any system in which participants can benefit from dishonest behaviour lacks the property of IC.

[2]There is no incentive to stop multiple selfish pools from emerging as can be seen in the real Bitcoin network which shows the presence of multiple mining pools [23]. We later show that the presence of multiple pools following the selfish-mine strategy causes the strategy to be less profitable.

step (0) in the state machine (SM). However, when a dishonest node mines a block (SM step (1)), the block is only announced to other dishonest nodes in the selfish pool. At this stage, the honest nodes continue to mine on the older public block n_p while the dishonest nodes mine on the newer secret block with serial number $n_s = n_p + 1$. We refer to the segment of the blockchain at the dishonest nodes comprising of the secret block(s) n_s as the *secret extension*. As long as the selfish pool has not learned[3] of any competing public block (block at the same depth of the blockchain as n_s), the pool assumes a lead of one block and continues to mine on the secret block n_s hoping to achieve a lead of two of more blocks. This is presented in lines 8 to 14 of Algorithm 1. However, since the pool is in the minority, the honest nodes will catch up.

When an honest node mines a block, it immediately broadcasts it to all the nodes in the network. When the block is received by the pool [4], the pool computes the *lead* and a course of action is selected based on the value of the *lead*. The *lead* refers to how long the secret branch is compared to the public branch of the blockchain. The secret branch comprises the secret extension and the parent public branch (the chain of blocks beginning at the predecessor to the first secret block all the way to the genesis block). This is presented in lines 17 to 36 of Algorithm 1. If the last public block known to the selfish pool has serial number n_p and latest secret block n_s, then

$$lead = n_s - n_p. \tag{5.1}$$

If $lead < 0$, it means that the secret branch has fallen behind the public branch. The selfish pool could continue mining on the last block n_s of the secret extension, but the probability of catching up with the majority of honest nodes is negligible, so the selfish pool abandons the secret extension and starts mining on the last known public block n_p.

If $lead = 0$, it means that the secret branch and the public branch are of equal length. The selfish pool immediately publishes (making a secret block public by broadcasting it to all the honest nodes) the only secret block in the extension causing the blockchain to fork (after a transmission delay) at all the honest nodes and a race to begin as to which branch will form the main chain. This is presented in lines 24 to 28 of Algorithm 1 and step (8) of the SM. Depending on the network topology and propagation delays, one of the two competing blocks arrives first at an honest node in the network. Since the honest nodes follow the Bitcoin protocol rules, they will mine on the block they first received whether it is a public block or a published block. Consequently, some honest nodes will begin mining on the recently published

[3]In a network model characterized by block propagation delays such as investigated in this thesis, a competing block may have been discovered but delayed in reaching the selfish pool.

[4]The selfish-mine algorithm may be executed only at a single controller (gatekeeper) or by every dishonest node. In this sense, the term pool is used to refer generally to both cases, centralized and distributed execution of the selfish-mine strategy.

block hence extending the formerly secret branch. We refer to the fraction of honest nodes that mine on the published block as γ, which is the measure of how often a published block arrives first at an honest node when there is a race. An analysis of the effect of block propagation delays on γ using a three node network configuration is presented in Appendix B. Further discussion on γ can be found in [34, 35]. At the same time, nodes in the selfish pool continue mining on the published block. Eventually, a new block is mined that extends either the public branch or the formerly secret branch. There are three possible outcomes.

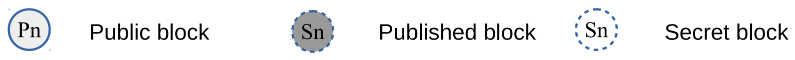
1. The selfish pool mines a successor block to the published block n_s. This is shown in Figure 5.1b and step (9) of the SM. In this case, the pool wins the race and receives the reward for both blocks. Both the honest and dishonest nodes begin to mine on the winning block.

2. An honest node mines a successor block to the published block n_s. This is shown in Figure 5.1c and step (10) of the SM. In this case, the pool still wins the race but is only credited for one block, the predecessor to the winning block. Both the honest and dishonest nodes begin to mine on the winning block.

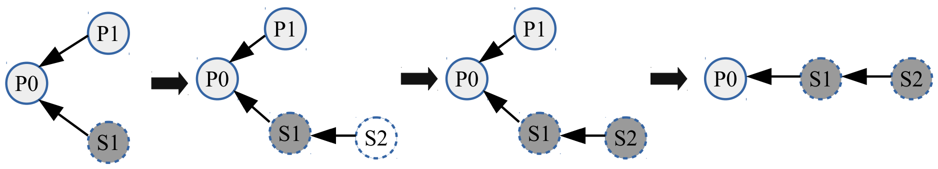
3. An honest node mines a successor block to the competing public block n_p. This is shown in Figure 5.1d and step (10) of the SM. In this case, the pool loses the race and the revenue for block n_s while the honest nodes are credited for both public blocks. The pool abandons the secret extension and both the honest and dishonest nodes begin to mine on the winning block.

If $lead = 1$, it means the secret branch is one block longer than the public branch. To ensure that the secret blocks end up being part of the main chain, the pool publishes both secret blocks in the secret extension as shown in Figure 5.2b. Both blocks reach the honest nodes and the honest nodes abandon mining on the block they were mining on. This is presented in lines 29 to 32 of Algorithm 1 and step (7) of the SM.

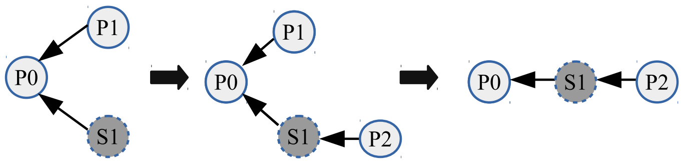
If $lead > 1$, the pool has a comfortable lead over the honest nodes. The pool publishes the first unpublished block in the extension as shown in Figure 5.3b. This is presented in lines 29 to 32 of Algorithm 1 and step (5) of the SM. Step (4) of the SM represents the case where the pool mines a block when $lead > 1$.



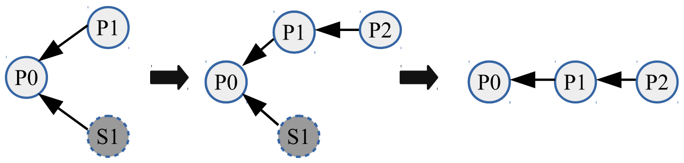
(a) Legend.



(b) Outcome 1: the selfish pool wins the race.

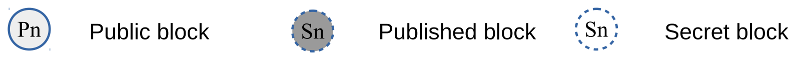


(c) Outcome 2: the revenue is shared.

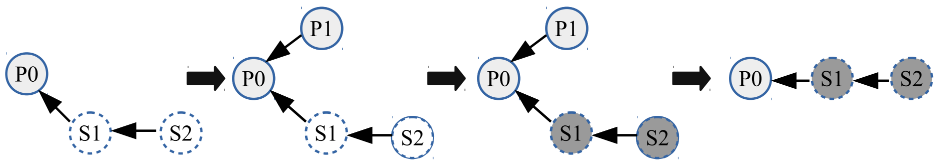


(d) Outcome 3: an honest miner wins the race.

Figure 5.1: Possible outcomes of a race at a dishonest node.



(a) Legend.



(b) Lead reduced to 1, both secret blocks are published.

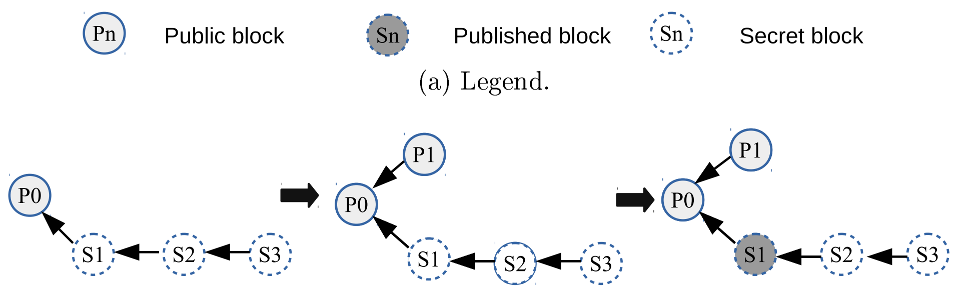
Figure 5.2: Both secret blocks are published at a dishonest node following the announcement of a public block.

Algorithm 1 The selfish-mine algorithm.

```
1: procedure INITIALIZE
2:     blockchain ← publicly known blocks
3:     secretExtension ←empty
4:     race ← FALSE
5:     mine on the last block of the blockchain $n_p$
6: end procedure

7: procedure SECRETMINE(block)
8:     append block to secretExtension
9:     if race =TRUE then
10:        publish block
11:        race ←FALSE
12:        secretExtension ←empty
13:    end if
14:    mine on block
15: end procedure

16: procedure HONESTMINE(block)
17:    append block to blockchain
18:    $n_p \leftarrow n_p + 1$                    ▷ the last public block has serial $n_p$
19:    $lead \leftarrow n_s - n_p$      ▷ the last block on secretExtension has serial $n_s$
20:    if $lead < 0$ then
21:        race ←FALSE
22:        secretExtension ←empty
23:        mine on block $n_p$
24:    else if $lead = 0$ then
25:        race ←TRUE
26:        publish the only block $n_s$ in the secretExtension
27:        secretExtension ←empty
28:        mine on block $n_s$
29:    else if $lead = 1$ then
30:        publish all two blocks in the secretExtension
31:        secretExtension ←empty
32:        mine on block $n_s$
33:    else
34:        publish the first unpublished block in the secretExtension
35:        mine on block $n_s$
36:    end if
37: end procedure
```



(a) Legend.

(b) Tit for tat publishing.

Figure 5.3: The first secret block is published following the announcement of a public block.

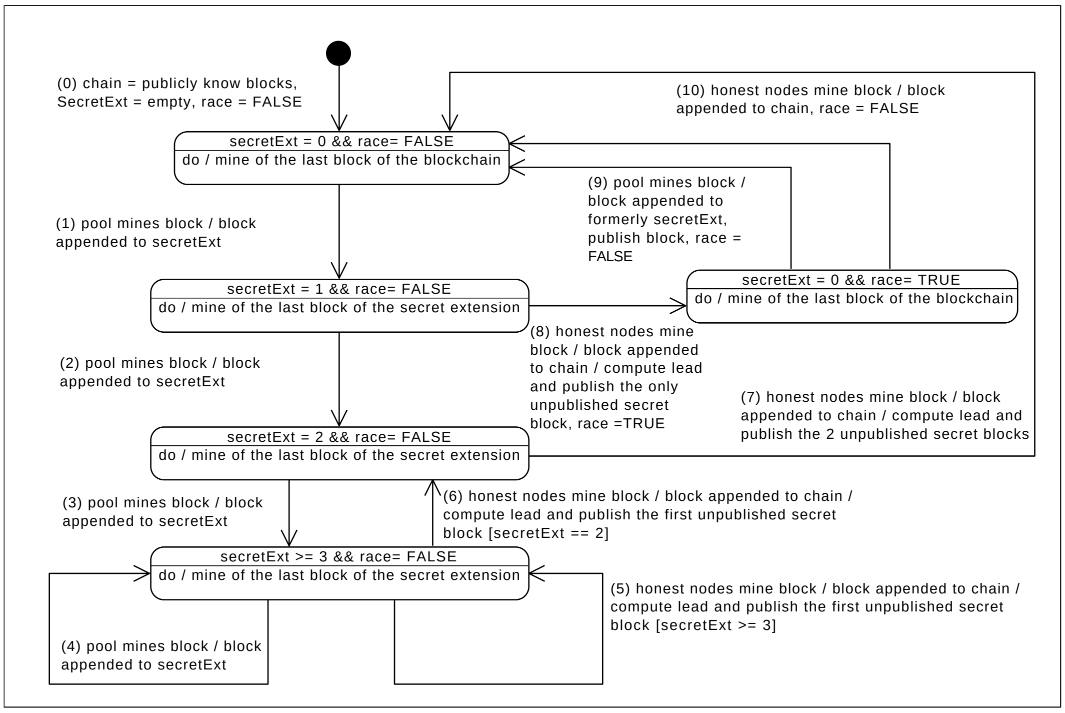

(0) chain = publicly know blocks, SecretExt = empty, race = FALSE
(10) honest nodes mine block / block appended to chain, race = FALSE
secretExt = 0 && race= FALSE
do / mine of the last block of the blockchain
(9) pool mines block / block appended to formerly secretExt, publish block, race = FALSE
(1) pool mines block / block appended to secretExt
secretExt = 0 && race= TRUE
do / mine of the last block of the blockchain
secretExt = 1 && race= FALSE
do / mine of the last block of the secret extension
(8) honest nodes mine block / block appended to chain / compute lead and publish the only unpublished secret block, race =TRUE
(2) pool mines block / block appended to secretExt
(7) honest nodes mine block / block appended to chain / compute lead and publish the 2 unpublished secret blocks
secretExt = 2 && race= FALSE
do / mine of the last block of the secret extension
(3) pool mines block / block appended to secretExt
(6) honest nodes mine block / block appended to chain / compute lead and publish the first unpublished secret block [secretExt == 2]
secretExt >= 3 && race= FALSE
do / mine of the last block of the secret extension
(5) honest nodes mine block / block appended to chain / compute lead and publish the first unpublished secret block [secretExt >= 3]
(4) pool mines block / block appended to secretExt

Figure 5.4: The selfish-mine state machine.

5.2 Evaluating the Selfish-Mine Strategy

5.2.1 Evaluation Criteria

In order to evaluate the profitability of the selfish-mine strategy, we consider two approaches, mining efficiency, and relative revenue.

5.2.1.1 Mining Efficiency

We consider the *mining efficiency* to be a relative measure of how well off a group of dishonest miners is doing as compared to the honest miners. Let N'_p denote the total number of blocks mined by the honest nodes and let $N_p \leq N'_p$ denote the total number of blocks mined by the honest nodes that end up in the main chain. The honest miners without doubt have received the revenue due for mining N_p blocks. Similarly, let N'_s denote the total number of blocks mined by the selfish pool and $N_s \leq N'_s$ denote the total number of blocks mined by the selfish pool that end up in the main chain. The dishonest miners without doubt have received the revenue due for mining N_s blocks. We define the *public mining efficiency* (P_p) as

$$P_p = N_p/N'_p, \tag{5.2}$$

and *pool mining efficiency* (P_s)

$$P_s = N_s/N'_s. \tag{5.3}$$

These equations express the outcome of blocks credited per effort (blocks mined) for the honest and dishonest nodes respectively. We now define the *mining efficiency* D as

$$D = P_s/P_p. \tag{5.4}$$

From this definition, the selfish pool is performing better than the honest nodes if $D > 1$. At this point, the pool's utility (reward per effort) is higher than that of the honest nodes since the pool is losing fewer blocks per block mined by the pool.

5.2.1.2 Relative Revenue

Eyal and Sirer [32] use the relative revenue to measure how profitable the selfish-mine strategy is. Let N_s and N_p denote the total number of blocks mined by the selfish pool and the honest nodes respectively that end up in the main chain at the end of a mining period. The relative revenue is the fraction of blocks mined by a group of miners (either selfish or honest) that end up in the main chain during that mining period. Specifically, we define the relative pool revenue R_p as

$$R_p = N_s/(N_s + N_p), \tag{5.5}$$

and relative honest revenue R_o as

$$R_o = 1 - R_p. \tag{5.6}$$

Selfish mining is profitable if $R_p > \alpha$ where α is the fraction of the total computational power of the network that is owned by the selfish pool. The fraction α is representative of the pool's fair share [32] of the mining revenue.

5.2.2 Simulations and Results

We performed simulation experiments based on the model presented in Chapter 4 Section 4.1.1, but with a network of 40 nodes. Once again 12 replications were executed for each experiment and 95% confidence intervals were computed. A subset of the miners commanding $0 \leq \alpha \leq 0.5$ of the total computing power of the network follows the selfish-mine strategy. We focused our investigation on the following key aspects of the selfish-mine strategy as presented in [32].

1. Using a network model that takes block propagation delays into account. We make a comparison between the performance of the pool when there is instantaneous communication (Eyal and Sirer's model [32]) and when all communication is subject to delays. Throughout this thesis, unless otherwise stated, our network model is characterized by a 10 seconds average communication delay[5].

2. The threshold at which it becomes profitable, in terms of the relative revenue and/or mining efficiency, for the pool to engage in selfish mining under the conditions in 1 above. [32] claims that an arbitrarily small pool of colluding miners can subvert the protocol.

3. The presence of multiple pools following the selfish-mine strategy. Though out this thesis, except in Section 5.2.2.4, we assume the presence of a single selfish pool in all our experiments.

In all cases, our evaluation of the performance of the pool(s) is based on mining efficiency and relative revenue.

5.2.2.1 Selfish Mining Under Communication Delays

The network model used to evaluate the selfish-mine strategy as presented in [32] does not take block propagation delays into account. Our network model, on the other hand, does not assume instantaneous communication. In this section, we make a comparison of the performance of the pool using the two network models. In order to examine the effect of block propagation delays on the performance of the pool, we ran several simulation experiments while varying the average communication delay in the network $t, 0 \leq t \leq 20$ seconds. The results are summarized in Figures 5.5 and 5.6.

Figure 5.5 shows the relative pool revenue R_p for $t \in \{0, 1, 10, 20\}$ seconds. Figure 5.6 is a pseudo 3D plot of the *mining efficiency* obtained from simulation experiments. The data in Figure 5.6 should be read with the key on the side representing the color codes for the different values of the *mining*

[5]In a real network, the block transmission delay is a function of the block size and latency on the link. However, for our simplistic network model, the block transmission delay and the communication delay are synonymous and used interchangeably.

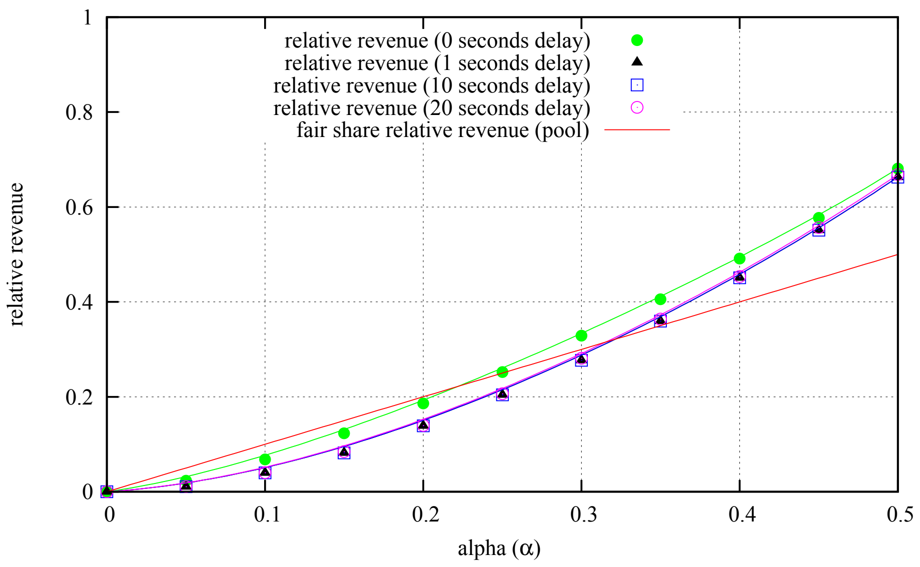


Figure 5.5: The relative revenue in a network with an average communication delay t, $t \in \{0, 1, 10, 20\}$ seconds.

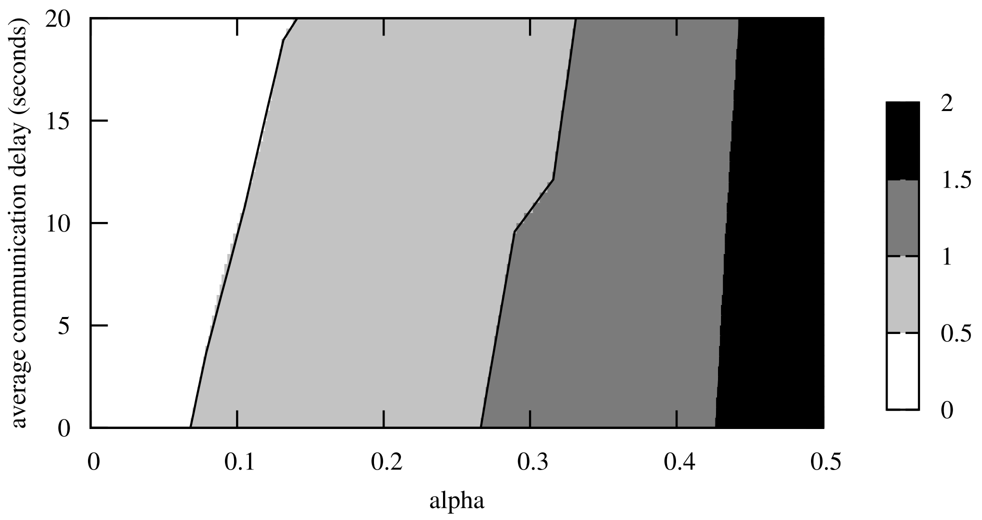


Figure 5.6: Mining efficiency in a network with an average communication delay t, $0 \leq t \leq 20$ seconds.

efficiency D. In both cases, the dishonest nodes act as a single pool. From the figures, we make the following observations.

- When there is instantaneous communication in the network (0 seconds delay), the threshold at which selfish mining becomes profitable is lower than when there are communication delays in the network. Figure 5.6 shows that the selfish-mine strategy is profitable, in terms of the mining efficiency ($D > 1$), for $\alpha \approx 0.26$. Figure 5.5 complements this value by showing that the profitability threshold for selfish mining, in terms of the relative pool revenue R_p, is $\alpha \approx 0.22$.

- When communication delays are introduced in the network, the threshold at which selfish mining becomes profitable is higher ($\alpha > 0.3$) than when there are no communication delays in the network as shown in Figure 5.5. From Figure 5.6, when $\alpha \geq 0.33$, selfish mining is profitable regardless of the block transmission delay.

- From Figure 5.5, the data points[6] representing the performance of the pool in a network with an average communication delay of 1, 10, and 20 seconds are almost superimposed on each other. This means that when there are communication delays in the network, it is immaterial whether the average delay is 1 or 20 seconds, the performance of the pool, in terms of the relative pool revenue R_p, is almost identical.

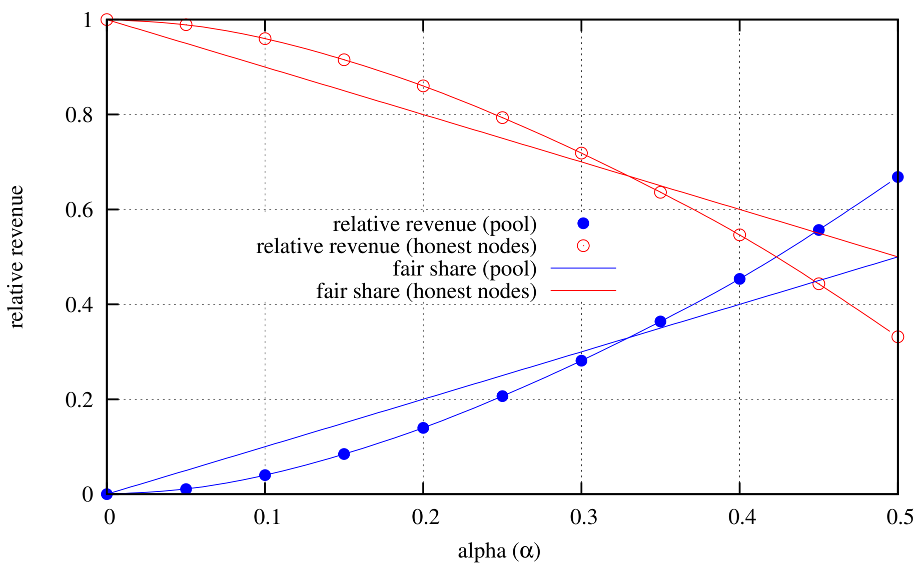


Figure 5.7: The relative revenue of the pool and the honest nodes in a 40 node network with an average block transmission delay of 10 seconds.

In addition, the relative revenues of the selfish pool and the honest nodes for the case where the average block transmission delay was 10 seconds (our

[6]Throughout this thesis, the lines in the graphs do not represent actual measurements from simulations. The data points represent experimental measurements. The lines are only used to improve the readability of the data.

delay model) were computed and the results are presented in Figure 5.7. From the graph, we can see that the pool begins to earn more than its fair share when $\alpha > 0.3$. The results agree with those computed using mining efficiency as a measure of profitability.

We conclude that in order to profit from selfish mining, at least in terms of R_p and D, the pool cannot be arbitrarily small. This is expected since selfish mining is a gamble and the probability of a small pool winning races and/or extending the secret extension to be significantly long, is small (see Section 5.2.2.2). We showed in [34] that for small pools ($\alpha < 0.3$), the honest nodes win the majority of the races. In addition, Eyal and Sirer [32] manipulate γ as wished, making it possible even for small pools to win all races when $\gamma = 1$. In our model, γ is an observation unless otherwise stated, as is the case in Section 5.2.3.3. Our results in Figure 5.7 agree with those of Eyal and Sirer [32] for $\gamma = 0.5$. We further showed in [34] that in a large network, for the case where the selfish pool is managed by a single node, the gatekeeper (see Section 5.2.2.3), γ is low and $\gamma << 1$.

5.2.2.2 The Length of the Secret Extension

Figures 5.8 and 5.9 show the length of the secret extension for $\alpha \in \{0.4, 0.5\}$ and $\alpha \in \{0.475, 0.5\}$ respectively in a network with an average communication delay of 10 seconds. While the rest of the experimental results are based on a 70 day mining period, the two figures show the length of the secret extension at every mining event for the first the 35 days of simulated mining.

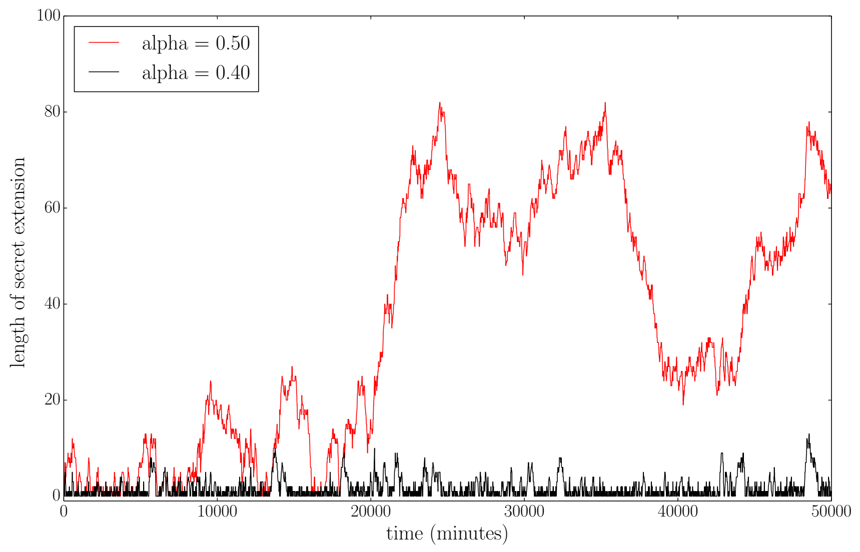


Figure 5.8: The length of the secret extension in a network with an average block transmission delay of 10 seconds for $\alpha \in \{0.4, 0.5\}$.

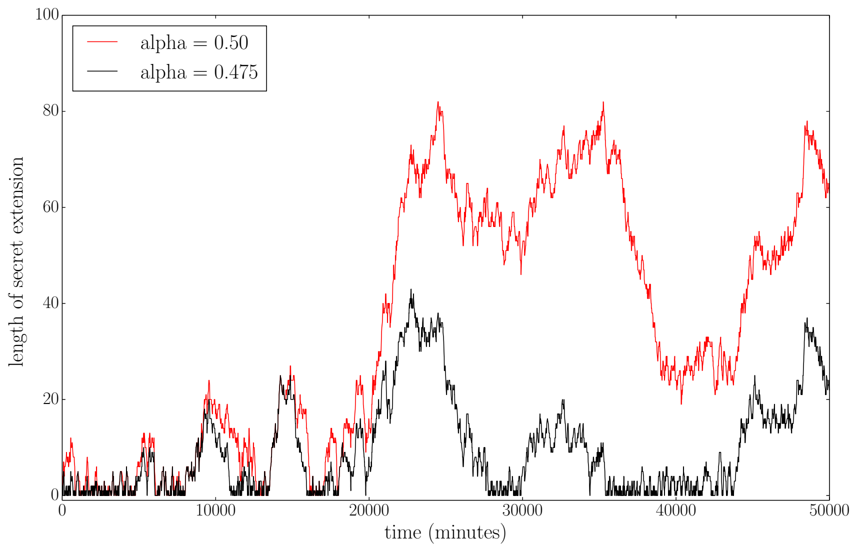


Figure 5.9: The length of the secret extension in a network with an average block transmission delay of 10 seconds for $\alpha \in \{0.475, 0.5\}$.

We make the following remarks about the growth of the secret extension.

- When $\alpha \leq 0.4$, the number of blocks in the secret extension is almost always less than 10 blocks as shown by the lower line in Figure 5.8.
- The probability of the selfish pool building a large secret extension is directly proportional to the size of the pool. As the pool increases in size, commanding a greater fraction of the total computational power of the network, the pool is able to grow its secret extension much longer. From Figures 5.8 and 5.9, the secret extension grows up to a maximum of about 15 blocks when $\alpha = 0.4$, 40 blocks when $\alpha = 0.475$, and 80 blocks when $\alpha = 0.5$.
- If $0 < \alpha < 0.5$ as defined by Equation 1.2, the honest nodes are bound to catch up to the pool even when the pool has a sufficiently large lead. Figure 5.9 shows that the secret extension grows to an extent of having 40 blocks when $\alpha = 0.475$, but the honest nodes catch up, reducing the lead of the pool. However, when the pool commands at least half the computational power of the network, once the pool builds a sufficient lead, the honest nodes may never catch up during the time period simulated. This is shown by the upper line in the two figures (when $\alpha = 0.5$, the selfish pool almost always has a lead of several secret blocks and the extension grows to a maximum of about 80 blocks).

In the generalized selfish-mine strategy discussed in Chapter 6, we introduce a parameter, minimum lead ml, that causes the pool to publish one secret block in reaction to the selfish pool discovering another secret block when the length of the secret extension equals ml. This ensures that the length of the secret extension never exceeds ml.

5.2.2.3 The Gatekeeper

According to [32], *it is immaterial whether the dishonest miners operate as a single group, as a collection of groups, or individually.*

We investigate whether there is a performance difference between a distributed and centralized implementation of the selfish-mine strategy. The selfish-mine strategy as implemented in [32] is based on the assumption that there exists a node in the pool acting as a controller, referred to as the gatekeeper. An illustration of such a configuration is presented in Figure 5.10.

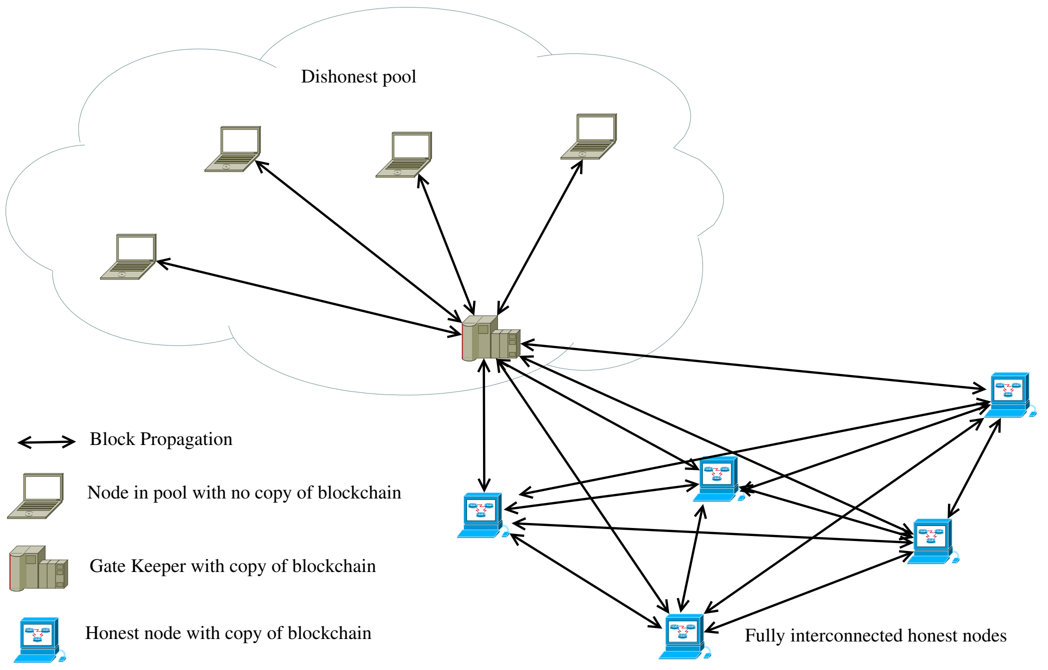

Dishonest pool
Block Propagation
Node in pool with no copy of blockchain
Gate Keeper with copy of blockchain
Honest node with copy of blockchain
Fully interconnected honest nodes

Figure 5.10: Pool with a gatekeeper.

The gatekeeper is a dishonest node in the pool that runs the selfish-mine algorithm. It is the only dishonest node visible to the honest nodes although the honest nodes are not aware that the gatekeeper is a dishonest node. The gatekeeper is the only dishonest node that receives announcements of blocks from the honest nodes[7] and sends published blocks mined by the pool to the honest nodes. Besides this, the gatekeeper maintains a copy of the blockchain

[7]It is possible to have variations where all nodes can receive announcements of public blocks and forward these to the gatekeeper. [32] suggests a Sybil attack that operates this way with additional non-mining nodes.

while other nodes in the pool only need to store a single block upon which to mine.

When a dishonest node mines a block B, it sends a copy of that block to the gatekeeper which decides what to do with the block B. Depending on the state of the blockchain, when block B arrives at the gatekeeper, an appropriate action is selected from below.

- The block is added to the blockchain. If there is a race, block B is immediately published to all the nodes in the network. When the nodes in the pool, besides the gatekeeper and the node that mined the block, receive the published block B, the nodes begin to mine on block B.

- Block B is discarded since it is outdated and the gatekeeper is aware of an alternative block B'. This is possible due to transmission delays. Node n which mined block B may not have received the message to mine on B'. The gatekeeper broadcasts to all nodes in the pool to mine on block B'.

The gatekeeper is also a miner. Mining at the gatekeeper is always efficient as it is aware at all times of which block to mine on. Other nodes in the pool may waste their effort mining on an outdated block due to transmission delays. All blocks found by the gatekeeper are immediately added to the blockchain and broadcast to the rest of the nodes in the pool.

Simulation experiments were conducted based on the model presented in Chapter 4 Section 4.1.1 but with 40 nodes, for the distributed and centralized implementation of the selfish-mine strategy. The results are shown in Figure 5.11.

The results in Figure 5.11 show that the performance of the centralized and the distributed implementation of the selfish-mine algorithm are almost identical for the model under investigation. We agree with [32] that it is immaterial whether the dishonest nodes act as individuals or as a group.

5.2.2.4 Two Selfish Pools

Since there is no incentive to stop multiple pools from following the selfish-mine strategy at the same time [23], we investigate the effect of having two selfish pools following the selfish-mine strategy. Each pool is approximately the same size and implements the selfish-mine algorithm as described in Section 5.1. However, each selfish pool does not know of the existence of the other selfish pool so they only forward secret blocks to nodes in their respective pools. We computed the combined mining efficiency and relative pool revenues of the two pools and the results are shown in Figures 5.12 and 5.13 respectively.

From Figure 5.12, we observe that selfish mining is profitable ($D > 1$) for pool sizes exceeding (combined $\alpha \approx 0.48$) when the average block transmission

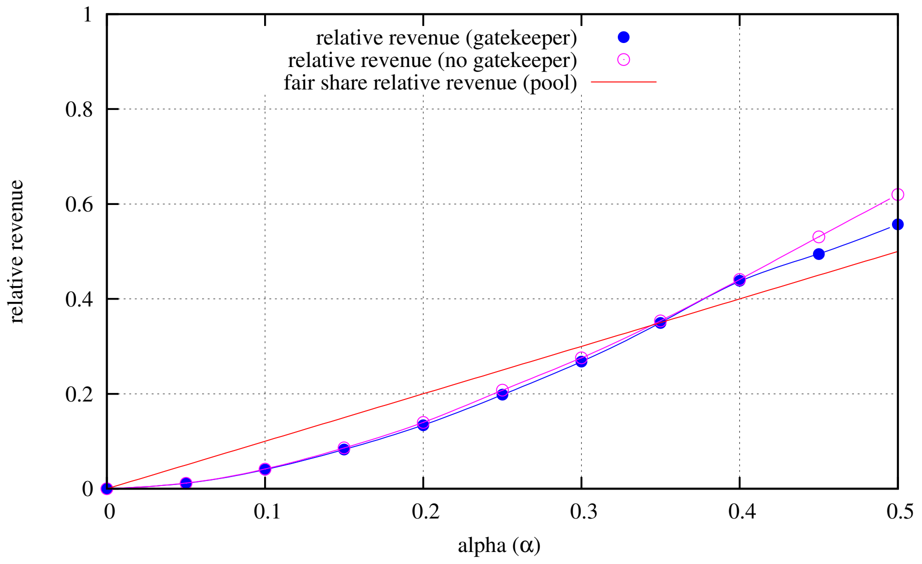


Figure 5.11: The relative revenue of the pool for a distributed and a centralized implementation of the selfish-mine strategy.

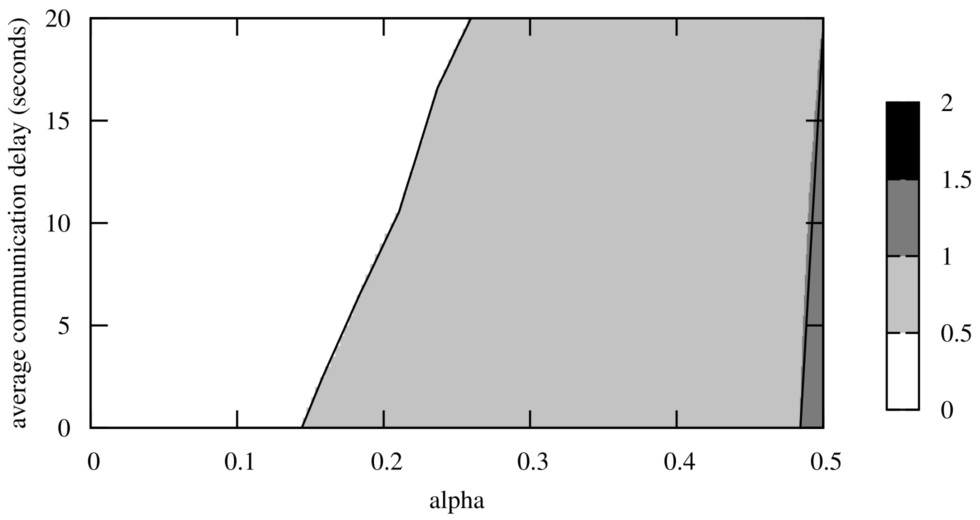


Figure 5.12: The combined mining efficiency of two pools in network with an average communication delay t, $0 \leq t \leq 20$ seconds.

delay is low but becomes unprofitable as the average block transmission delay increases.

Figure 5.13 shows that the combined relative revenue of the pools is less than their fair share for all pool sizes. Since the underlying principle in the

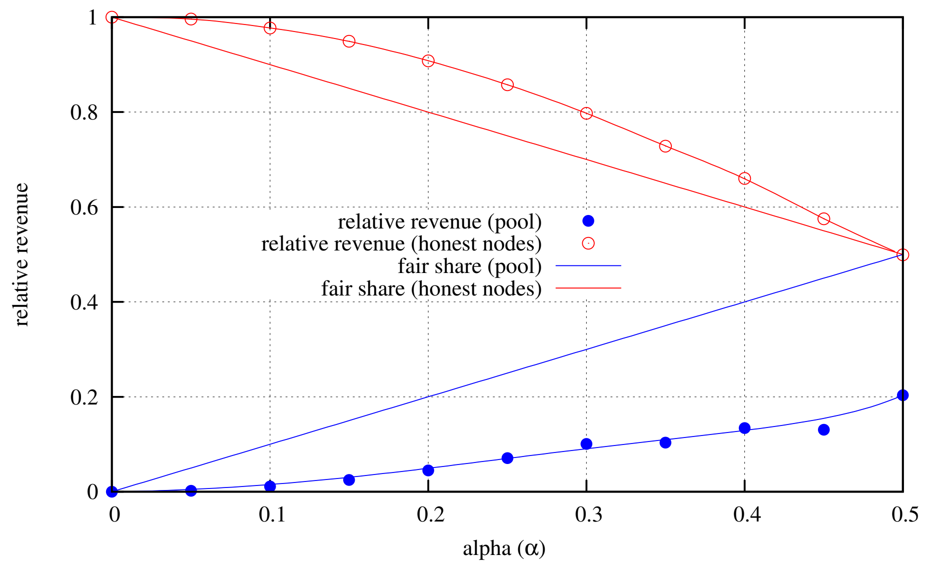


Figure 5.13: The combined relative revenue of two pools in network with an average communication delay t, $t = 10$ seconds.

selfish-mine strategy is secrecy and the pools do not share block information, they negate each other's efforts by revealing competing blocks [23]. Each pool not only competes with the honest nodes but with the other pool as well, effectively reducing their revenue. Intuitively, we can see that the greater the number of selfish pools (following the selfish-mine strategy) present in the network, the less the benefit in following the selfish-mine strategy.

5.2.3 Is the Selfish-Mine Strategy Profitable?

From Section 5.2.2.1, we saw that when block propagation delays are present, the selfish-mine strategy becomes profitable for $\alpha > 0.3$. However, profitability was measured in terms of the relative pool revenue and this does not necessarily mean that the pool incorporated more blocks in the main chain when following the selfish-mine strategy than it would have if it followed the Bitcoin rules. To elaborate on this, consider the case where the pool incorporates 100 blocks in the main chain while following the Bitcoin rules and it incorporate 80 blocks while following the selfish-mine strategy. The rest of the network incorporates 300 and 200 blocks respectively. When we compute the relative revenues, the pool earns 0.25 while following the Bitcoin rules and earns 0.286 while following the selfish-mine strategy. It appears as if the pool is doing better when following the selfish-mine strategy. However, the actual revenue earned from 80 blocks (2000 bitcoins) is less than the revenue earned from 100 blocks (2500 bitcoins) at the current coin base rate. Essentially, both the pool and the rest of the network are worse off than they would have been if there was no selfish mining present.

Miners are assumed to be rational and that the goal of a miner is to maximize its revenue within a mining period [32]. Consider the performance of the pool during a fixed period of mining T. Due to forking, blocks are deleted during the same period resulting in the pool receiving revenue for N_s blocks out of the N'_s blocks mined.

We conducted a set of simulation experiments for a network of 40 nodes with a mining period $T = 70$ days (10,000 blocks are mined if $M = 10$) and $0 \leq \alpha \leq 0.5$. Once again, 12 replications were executed for each experiment. In experiment 1, the selfish pool follows the selfish-mine strategy with the average block mining interval M of 10 minutes throughout the mining period T. In experiment 2, variable M is initially set to 10 minutes. In the real Bitcoin network, the average mining interval is adjusted every fortnight by adjusting the difficulty, which is the number of leading zeros in the message digest H, computed over a string comprising the hash of the parent block, the Merkle root, and the nonce (see Chapter 3 Section 3.3.2). In our simulator, we use a simplistic approach to model the difficulty and mining process. The time it takes to mine a block is a random number sampled from an exponential distribution with a rate of change R $(0 < R < 1)$ and mean $M = 1/R$.

Throughout this thesis, unless otherwise stated, $R = 0.1$ resulting in an average M of 10 minutes. Changing R changes M consequently, in our model, the process of changing the difficulty can be modelled by changing R. As opposed to the approach taken in [34] where the difficulty was adjusted at the beginning of an experiment and kept constant, in experiment 2, the difficulty was adjusted every 2016 blocks (by adjusting R) as is the case in the real Bitcoin network using the equation

$$R = R_{old} + ((T_{actual} - T_{expected})/M_{standard})/T_{expected} \tag{5.7}$$

where T_{actual} minutes is the time it took to mine 2016 blocks, $M_{standard} = 10$ minutes, and $T_{expected} = 20160$ minutes is the expected time to mine 2016 blocks. The adjusted M can be computed from R using Equation 5.8

$$M = 1/R. \tag{5.8}$$

Experiment 3 is used as a control experiment to assess the performance of the pool in experiment 1 and 2 and assumes a pool of the same size as in experiments 1 and 2 except the pool is following honest mining as specified by the Bitcoin rules instead of the selfish-mine strategy (experiments 1 and 2). The results for all three experiments are discussed in Section 5.2.3.2 while Section 5.2.3.1 only compares results from experiment 1 and 3.

5.2.3.1 Pool Mining Efficiency (P_s)

Figure 5.14 shows the *pool mining efficiency* (P_s) and the *public mining efficiency* (P_p) defined in Section 5.2.1.1 for experiment 1 and experiment 3.

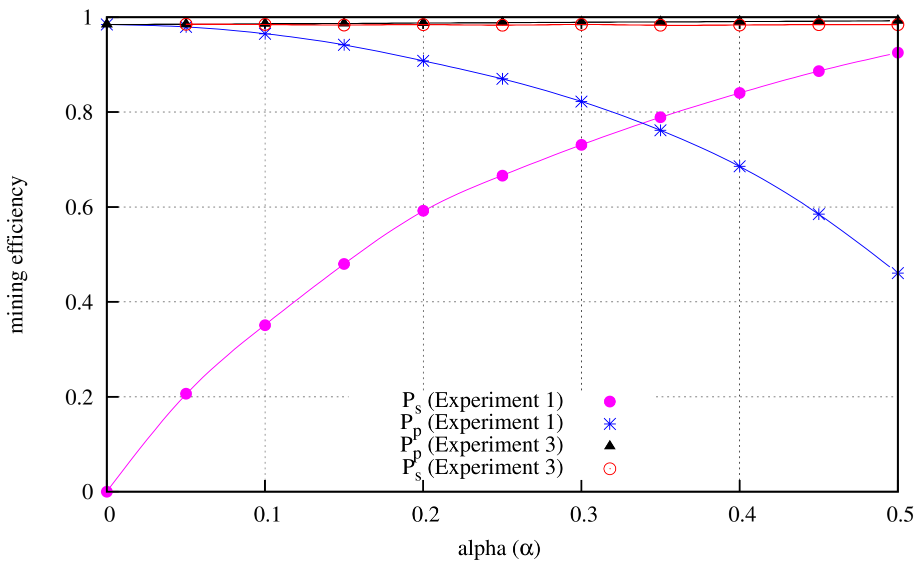


Figure 5.14: The performance of the pool (P_s) and the honest nodes (P_p) in a 40 node network.

It is clear from Figure 5.14 that when selfish mining is present (experiment 1), both the pool and the honest nodes are worse off than they would have been if no selfish mining was present (experiment 3). This is because the forking of the blockchain and deletion of blocks[8] that takes place when selfish mining is present causes the total block generation rate to be reduced (less than 6 blocks are added to the blockchain every hour) and consequently the overall revenue decreases. According to the Bitcoin specification, the block mining difficulty will be automatically adjusted downwards hence increasing the block generation rate to ensure that the blockchain grows one block longer every 10 minutes on average. The following section investigates the effect of adjusting the mining difficulty.

5.2.3.2 Adjusting the Block Mining Difficulty

Hitherto, our results have been based on experiments with a fixed average mining interval M. In the real Bitcoin network, the reduced overall revenue generation resulting from the presence of selfish mining would trigger a downward adjustment of the difficulty, hence reducing the average mining interval such that more than 6 blocks are mined every hour, on average, to compensate for the deleted blocks resulting from forks. The results of a comparison between the performance of the pool when M is fixed and when M is variable for $0.3 \leq \alpha < 0.5$ are presented in Table 5.1. We consider three experiments.

[8]Blocks are not deleted from the blockchain in the actual Bitcoin network. The transactions are removed from the block and added to the next block if they are not already incorporated in another block. The depleted block remains in the blockchain.

Table 5.1: Variable Mining Rate.

α	Experiment	Final M	Minimum M	N'_s	N_s	R_p
0.30	Experiment 1	10	10	3005	2197	0.2752
	Experiment 2	9.77	8.01	3325	2421	0.2743
	Experiment 3	10	10	2995	2949	0.2995
0.40	Experiment 1	10	10	4012	3371	0.4492
	Experiment 2	9.69	7.51	4562	3823	0.4495
	Experiment 3	10	10	4019	3953	0.3994
0.425	Experiment 1	10	10	4237	3649	0.4982
	Experiment 2	10	7.2	4870	4185	0.4979
	Experiment 3	10	10	4264	4197	0.4243
0.45	Experiment 1	10	10	4488	3979	0.5532
	Experiment 2	9.8	7.18	5219	4609	0.5527
	Experiment 3	10	10	4533	4463	0.4495
0.475	Experiment 1	10	10	4755	4317	0.6122
	Experiment 2	9.97	6.86	5546	5014	0.6097
	Experiment 3	10	10	4790	4712	0.4752

Experiment 1: selfish-mine where the difficulty is not adjusted. Experiment 2: selfish-mine where the difficulty is adjusted according to Equation 5.7. Experiment 3: all nodes are honest and the mining difficulty is kept constant throughout the experiment. Based on the results of the three experiments, we make the following observations.

- When M is constant, $M = 10$, the pool always performs worse, in terms of N_s, when following the selfish-mine strategy (experiment 1) than when following honest mining (experiment 3) for all values of α.

- For $\alpha < 0.45$, the pool mines more blocks (N'_s) when following the selfish-mine strategy (experiment 2) than when following honest mining (experiment 3) owing to the reduction of M introduced in experiment 2. However, despite the difficulty adjustments in experiment 2, the total number of blocks (N_s) that end up in the main chain is still less when following the selfish-mine strategy (experiment 2) than when following honest mining (experiment 3) even when $R_p > \alpha$. While varying M results in both the pool and the honest nodes mining more blocks than in experiment 1 (constant M), both the pool and the honest nodes end up losing more blocks due to the forking caused by selfish mining [34].

- When $R_p > 0.5$ ($\alpha \geq 45$), adjusting M results in the pool performing better when following the selfish-mine strategy than it would have when following honest mining. At this point, the pool incorporates more blocks into the main chain at the end of a mining period (N_s) when following the selfish-mine strategy (experiment 2) than when following honest mining (experiment 3). Following the selfish-mine strategy now pays off owing to the difficulty adjustments.

For every block in the main chain, a reward B is awarded to the miner. We define the selfish mining profit P during the mining period T as

$$P = BN_s - C \tag{5.9}$$

where C is the cost associated with operating the mining rigs during the period T. We assume that during period T, all n miners (honest and dishonest) are constantly involved in solving the cryptographic puzzle each at a cost c (our model assumes uniform hashing power for all miners hence the uniform costs c of operating each individual rig). During this time, the pool (collective cost for all miners in the pool) incurs a cost

$$C = c\alpha n \tag{5.10}$$

to mine a total of N_s' blocks where n is the total number of nodes (rigs) in the network. The cost C is independent of the strategy followed by the pool (honest or selfish) and the number of blocks mined N_s'. Since B[9] and C are constant, it is in the interest of the pool when $\alpha < 0.45$ to follow honest mining as this maximizes N_s and consequently P as shown in Table 5.1. This is for the case where the difficulty is adjusted to ensure that on average, the blockchain grows 6 blocks longer every hour as is the case in the real Bitcoin network. For the case where the difficulty is kept constant, selfish mining is unprofitable (in terms of P) for all values of α.

We further showed in [34] that the reduced difficulty not only results in increasing the overall revenue generation but also increases the number of forks. This can afford a means for the honest nodes to detect that some miners are being dishonest.

5.2.3.3 Network Superiority

The network superiority [4] ns is the ability of an attacker, in this case, the selfish pool, to quickly learn of any new public blocks and to quickly propagate its published blocks through the Bitcoin network. Eyal and Sirer [32] propose a strategy, the Sybil attack, that if effectively implemented grants the selfish pool total network superiority. When $ns = 1$, the pool is said to have total network superiority. In essence, this means that with the current Bitcoin implementation, the pool has the ability to influence the rate at which pool blocks are propagated through the network consequently, winning all races. In our simulation model, the communication delay between two nodes a distance μ apart is given by

$$delay = kAN(\mu, \sigma) \tag{5.11}$$

where $k = 1 - ns$, A is a scaling constant, and $N(\mu, \sigma)$ is a random number sampled from a normal distribution with mean μ and variance σ.

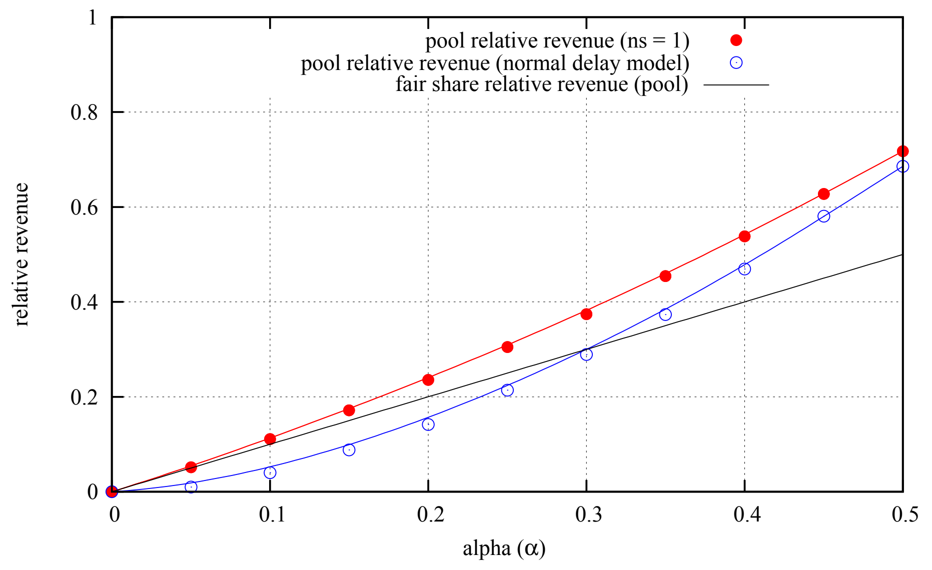


Figure 5.15: The relative revenue when the pool has total network superiority ($k = 0$).

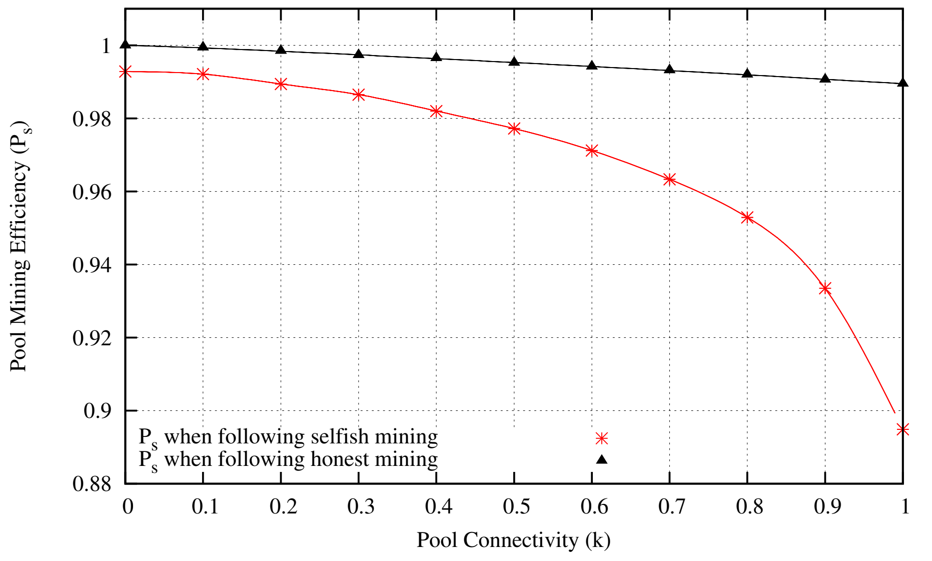


Figure 5.16: The performance of the pool (P_s) when following selfish and honest mining with varying k for $\alpha = 0.4$.

In order to investigate the effect of ns on the performance of the pool, we conducted a set of simulation experiments where the communication delay involving any dishonest node was subject to different values of k, $0 \leq k \leq 1$. When $ns = 1$, the pool has total network superiority. In this scenario ($k = 0$),

[9]The reward is halved every 210,000 blocks but is currently at 25 bitcoins per block.

the pool can propagate its published blocks instantly. On the other hand, communication between honest nodes was subject to delays (as expressed in Equation 5.11) with $k = 1$ in all experiments, representing the inability of the honest nodes to influence the propagation of their blocks through the network.

Figure 5.15 compares the relative pool revenue R_p when the pool has total network superiority, represented by the line labelled relative pool revenue ($ns = 1$), and when our usual delay model is used, represented by the line labelled relative pool revenue (normal delay model). When $ns = 1$, Figure 5.15 shows that the pool always performs better in terms of the relative revenue when following the selfish-mine strategy than when following honest mining for any pool size, α. This means that as far as relative revenue is concerned, the selfish-mine strategy is profitable for any arbitrarily small pool with total network superiority.

The performance of the pool in terms of the pool mining efficiency P_s when following honest mining and selfish mining for $0 \leq k \leq 1$ and $\alpha = 0.4$ is shown in Figure 5.16. Figure 5.16 shows that the pool loses more blocks when following the selfish-mine strategy than when following honest mining, hence the lower value of P_s for all values of k. It is interesting to note that even in the situation where the pool has total network superiority, $ns \approx 1$, our simulation model shows that the pool still loses more blocks when following the selfish-mine strategy as proposed by [32]. This further goes to show that it remains in the interest of the pool to follow honest mining.

5.3 Conclusion

Based on the results of our experiments, we make the following conclusions about the selfish-mine strategy.

1. The selfish-mine strategy as proposed in [32] seems to be a profitable strategy when evaluated in terms of the relative revenue and mining efficiency. However, we disagree with the assertion in [32] that an arbitrarily small pool can profit, in terms of the relative revenue, by following the selfish-mine strategy. Our simulation results show that the selfish-mine strategy is profitable, in terms of the relative revenue, for $\alpha > 0.3$. This holds true for a network model where we do not assume instantaneous communication or the ability of the selfish pool to achieve total network superiority [4] by implementing a Sybil attack [32] for example.

2. Block propagation delays affect the profitability of the selfish-mine strategy as opposed to the assumption in [32] where communication is instantaneous. Our simulation model is more faithful to the Bitcoin network since we take block propagation delays into account [26]. However, we observe that it is immaterial whether the average block transmission de-

lay in the network is 1 second or 10 seconds. The selfish-mine strategy yields identical results for all non-zero communication delays.

3. We further showed that any pool can profit from selfish mining as claimed in [32], in terms of the relative revenue, if the pool can achieve total network superiority. Although [32] gives proposals of how an attacker can achieve total network superiority, we are not convinced of the practicality of this attack in the real Bitcoin network.

4. We further showed that if the pool does not achieve network superiority, the selfish-mine strategy is only profitable (actual revenue earned) when $\alpha \geq 0.45$ only if the mining difficulty is lowered.

5. It is immaterial whether a distributed or centralized implementation of the selfish-mine strategy is followed by the selfish pool. The performance of the distributed and the centralized implementation is almost identical.

6. The presence of multiple pools following a the selfish-mine strategy is counter productive to each pool.

7. Above all, if miners are rational and wish to maximize their revenue (profit) from mining, it is in their interest to follow honest mining as this maximizes their profit P (actual revenue earned during a mining period) for $0 \leq \alpha < 45$.

Chapter 6

Genetic Optimization of Selfish-Mine

6.1 Genetic Algorithms

Genetic Algorithms (GAs) are often considered to be a form of artificial intelligence. They are an adaptive heuristic search technique based on Darwin's theory of evolution [51]. GAs are generally used in search and optimization problems. They perform a random yet intelligent search that is directed towards an optimal region within the search space by leveraging the history of the search. A GA is designed to mimic the processes of evolution in natural systems that exhibit survival of the fittest. The struggle for survival among individuals in such natural systems results in the weaker individuals being replaced by the fitter ones [75]. GAs exploit this property to ensure that the fittest (optimal) solution(s) to a problem is (are) found.

The focus of research in GAs has been robustness [37]. The implications of robustness for many systems are numerous. For example, redesigns which are usually costly may be avoided if systems can be made more robust. Robustness also entails that existing systems are enabled to better perform their functions if higher levels of adaptation can be achieved.

GAs have been applied in many areas from engineering to business, addressing routing optimization in telecommunication networks, computer gaming, engineering design, robotics, optimization of analysis of chemical kinetics and finance and investment strategies to mention a few examples.

6.1.1 Overview

A typical genetic algorithm requires,

- a genetic representation of the solution domain and
- a fitness function to evaluate the solution domain.

Usually, a candidate solution is represented as an array of bits [51]. The use of different types of arrays and structures can be employed in essentially the same way. The following key terms are defined.

- Generation: a population of n individuals.
- Individual: a point in the search space representing a possible (candidate) solution to the problem, coded as a fixed length vector of variables. An individual is analogous to a chromosome and the variables to genes, thus an individual's composition is comprised of several genes.

GAs solve a problem by enacting the survival of the fittest among n individuals [75]. A fitness score is assigned to each individual. The fitness score is used to measure how good (a solution to the problem) the individual is. The GA's objective is to find the individual(s) with the optimal (or near optimal) fitness score(s). To achieve this, the GA evolves generations of individuals.

Selective breeding is applied by selecting individuals (parents) with higher fitness scores to reproduce. Consequently, individuals (parents) with superior ranking are given more opportunities to produce descendants so that some of the characteristics from each parent are passed on. Since only n individuals are maintained per generation, room must be made for new arrivals, consequently individuals (with lower fitness scores) are removed and replaced by the new individuals. In this way, in succeeding generations, more desirable solutions will dominate.

Iteratively, new generations are produced comprising individuals with more desirable characteristics on average, and it is expected that this will be the consistent trend for all successive generations. In the long run, once individuals in each successive generation are no different (in terms of fitness) from previous generations, the algorithm is said to have converged.

6.1.2 Implementation Details

The evolution of the GA usually starts from a generation of randomly generated individuals. After the initial generation, the algorithm evolves through the application of following operators.

- **Selection**: This is how survival of the fittest is implemented. Individuals are evaluated using a fitness function and the fitter individuals are given preference, allowing them to pass on their genes to the next generation[1]. The fitness function can be implemented as an objective function to be maximized (or minimized) or by a subjective judgment.

[1]Evaluating the fitness of an individual can be very time-consuming depending on the fitness function, consequently certain selection methods evaluate the fitness of each individual while other methods evaluate only a random sample of the population.

- **Crossover**: Mating between individuals is implemented as a crossover function. This is the prime factor that distinguishes GAs from other optimization techniques. Consider two individuals, $S1 = 000111$ and $S2 = 101010$ which are chosen from the population using the selection operator. A random point along the strings is chosen. This is referred to as the crossover site. The strings are split at the crossover point and the resulting sub-strings are used to generate new individuals such that each individual is composed of two sub-strings, one from each parent. The two individuals that would result in crossing $S1$ and $S2$ if the crossover point is 2 are $S1' = 001010$ and $S2' = 100111$. These are added to the next generation and it is likely that fitter individuals are created by this process of recombining portions of good individuals.
- **Mutation**: Mutation introduces random modifications to individuals. A portion of the new individuals has some of their bits flipped with a low probability. The goal of mutation is to maintain diversity within the population and inhibit premature convergence [75].

As expected, each operator has its strengths and weakness. For example, using selection alone will tend to fill the population with copies of the best individuals from the population while the selection and crossover operators will tend to cause the algorithm to converge on a sub-optimal solution. For the best results, a combination of all operators is used. The following outlines how a typical GA would operate [75].

1. Begin by randomly initializing a $population(t), t = 0$.
2. Evaluate the fitness of each individual in the $population(t)$ using the fitness function.
3. Repeat the following until the termination criteria are satisfied.
    - Use the selection operator to select highly ranked individuals from the $population(t)$ to be parents.
    - Apply the crossover function on parents creating $population(t+1)$.
    - Apply the mutation operator on $population(t+1)$.
    - Evaluate the fitness of each individual in $population(t+1)$.

Several criteria exist for terminating a GA. The usual ones are listed below.

- An individual is found that satisfies the minimum fitness threshold.
- The maximum number of generations to evolve is reached.

- The algorithm converges (new individuals are no different from those in previous generations).
- A combination of the above.

6.2 Application to Selfish-Mine

6.2.1 General Selfish-Mine

In order to apply GA optimization to the selfish-mine strategy, we begin by specifying a generalized implementation of the selfish-mine algorithm [36]. According to [4], selfish-mine is a specialized case of a greater class of block discarding attacks referred to as st_k, $k = 0, 1, 2, \ldots, \infty$, where st_0 is honest mining, st_1 selfish-mine as proposed in [32] and st_k, $k > 1$ are variations of selfish-mine claimed to be more profitable than st_1, which is more profitable than st_0.

The general selfish-mine algorithm represents a greater variety of block discarding attacks than st_k. A specification of the general selfish-mine algorithm [36] is presented in Algorithm 2 based on Algorithm 1 in Chapter 5 and its operation is determined by the parameter vector $v = (ml, p_2, p_3, d_2, d_3, d_{sl}, ra)$ described below.

- ml is an integer such that $ml \geq 0$. ml denotes the minimum lead (length of the secret extension) required by the pool in order to publish one secret block in reaction to the pool discovering another secret block. The values of ml are interpreted as follows.

  $ml = 0$ means that blocks are never published except in reaction to the pool learning of a new public block. This is the case for the selfish-mine strategy as proposed by [32].

  $ml = 1$ means that the pool immediately publishes secret blocks once they are discovered. This is representative of honest mining.

  $ml > 1$ means the pool publishes one block when the secret extension is ml long. This may be necessary when the pool is large in order to limit the length of the secret extension.

- p_2, p_3 are integers. They denote the number of blocks the selfish pool reveals in reaction to the pool learning of a new public block when $lead = 2$ or $lead \geq 3$ respectively. The $lead$ is defined as $lead = n_s - n_p$ where n_s and n_p are the serial numbers (depth) of the last secret block and public block known to the pool respectively.

  In a network characterized by communication delays, it may be advantageous to publish more blocks than recommended by [32]. This is because public blocks may have been announced but are delayed in reaching the pool due to communication delays. Publishing more than one block ensures that the secret blocks have a high chance of being accepted by the honest nodes. This compensates for public blocks that may still be in transit [36].

- d_2, d_3, d_{sl} are integers such that $0 \leq d_2, d_3, d_{sl} \leq 1200$ seconds. These values denote the pool's delays in publishing a block except when there is a race or *lead* is reduced to one. In such cases, the secret block is published immediately. In all other cases, the pool delays publishing a block by d_2, d_3 seconds in reaction to the announcement of a public block when the $lead = 2$ or $lead \geq 3$ respectively. The delay d_{sl} seconds is used for publications in reaction to the pool discovering another secret block. The d_{sl} parameter is only used when $ml > 0$.

  The idea behind these delays is to deny the honest nodes the opportunity to mine on the block that will form part of the main chain. This increases the chances of the pool being rewarded revenue for the next successive secret blocks.

- ra is an integer that denotes whether the pool will engage in a race or not when *lead* is reduced to zero.

  $ra = 0$ means the pool does not engage in a race when *lead* is reduce to zero. Instead, the pool immediately abandons the secret extension and begins mining on the block in the public chain. This may be necessary when the pool contains a small number of nodes and the chances of winning the race are minimal as the pool may waste computational effort mining on the formerly secret block.

  $ra = 1$ means the pool engages in a race when *lead* is reduced to zero. This is the case for the selfish-mine strategy as proposed by [32].

The general selfish-mine algorithm includes honest mining given by $v = (1, \square, \square, \square, \square, 0, 0)$ and the selfish-mine strategy as proposed by [32] given by $v = (0, 1, 1, 0, 0, 0, 1)$ where $\square$ represents an inapplicable parameter. Equipped with the above, we can now use a GA to find an optimal configuration of the parameter vector v that yields near maximal revenue for the selfish pool.

Algorithm 2 The general selfish-mine algorithm.

```
 1: procedure INITIALIZE
 2:     read $ml, p_2, p_3, d_2, d_3, d_{sl}, ra$
 3:     $secretExtension$ ←empty
 4:     $race \leftarrow$ FALSE
 5:     mine on the last block of the chain
 6: end procedure

 7: procedure SECRETMINE($block$)
 8:     append $block$ to $secretExtension$: $n_s = n_s + 1$
 9:     $lead \leftarrow n_s - n_p$        ▷ The last block of $secretExtension$ has serial $n_s$
10:     if $race$ = TRUE and $ra = 1$ then
11:         PUBLISH$(1, 0)$
12:         $race$ ←FALSE
13:         $secretExtension$ ←empty
14:     else if $lead \geq ml > 0$ then
15:         PUBLISH$(1, d_{sl})$
16:     end if
17:     mine on $block$
18: end procedure

19: procedure HONESTMINE($block$)
20:     append $block$ to $blockchain$
21:     $n_p \leftarrow n_p + 1$                    ▷ the last public block has serial $n_p$
22:     $lead \leftarrow n_s - n_p$        ▷ the last block of $secretExtension$ has serial $n_s$
23:     if $lead < 0$ or ($lead = 0$ and $ra = 0$) then
24:         $race$ ←FALSE
25:         $secretExtension$ ←empty
26:         mine on block $n_p$
27:     else if $lead = 0$ and $ra = 1$ then
28:         $race$ ←TRUE
29:         PUBLISH$(1, 0)$
30:         mine on block $n_s$
31:     else if $lead = 1$ then
32:         PUBLISH$(2, 0)$
33:         mine on block $n_s$
34:     else if $lead = 2$ then
35:         PUBLISH$(p_2, d_2)$
36:         mine on block $n_s$
37:     else
38:         PUBLISH$(p_3, d_3)$
39:         mine on block $n_s$
40:     end if
41: end procedure

42: procedure PUBLISH(integer $n$, integer $delay$)
43:     remove $n$ blocks from $secretExtension$
44:     publish $n$ blocks after $delay$ seconds
45: end procedure
```

6.2.2 Simulations and Results

We extended our simulator to use a C++ Genetic Algorithm Library, GAlib, developed at MIT [2] to search for an optimal configuration of the parameter vector v that yields better performance for the selfish pool. GAlib was selected for the following reasons.

- It is one of the popular libraries available for C++ genetic programming.
- It is fairly easy to set up and has extensive documentation and examples.
- It is object-oriented making it easy to extend.
- Most importantly, the default operators available in the library were sufficient for our application.

We performed a GA search using rank-based selection. In the first set of experiments, we maximized the relative pool revenue R_p. In the second set of experiments, we maximized the number of confirmed blocks (the total number of blocks mined by the pool that end up in the main chain at the end of the simulation) N_s. While the general trend is to examine selfish-mine based on the relative revenue [4, 32, 63], we considered both R_p and N_s since we showed in Chapter 5 Section 5.2.3 that having a higher relative revenue does not necessarily entail more confirmed blocks in the main chain. In addition, the parameter vector v was constrained as follows: $1 \leq ml \leq 15$, $1 \leq p_2, p_3 \leq 3$, and $0 \leq d_2, d_3, d_{sl} \leq 1200$. Other details of the GA configuration are as follows.

- The population size (individuals per generation) $popSize = 40$.
- The mutation rate $pmut = 5\%$.
- The crossover rate $pcross = 50\%$.
- The convergence threshold was set to 99%.
- The GA looked back $nconvergence$ generations to check for convergence ($nconvergence = 20$).
- The GA evolved a maximum of $ngen$ generations before termination ($ngen = 100$).
- The GA search terminated when either the algorithm converged or $ngen$ generations were evolved.

A summary of how our GA is implemented is given in Algorithm 3.

[2]This research was performed using GAlib, a library of genetic algorithm components (http://lancet.mit.edu/ga/).

Algorithm 3 The genetic algorithm.

```
1: procedure GA($ngen, popSize, pcross, pmut, nconvergence$)
2:     $benchmarkScore \leftarrow$ score of standard selfish-mine
3:     $gen \leftarrow 0$
4:     $converged =$ FALSE
5:     randomly generate $popSize$ genomes $G(gen)$
6:     while $gen < ngen$ and $converged =$ FALSE do
7:         for all $g \in G(gen)$ do
8:             $g(rank) \leftarrow$ OBJECTIVEFUNCTION($g$)
9:         end for
10:        select $pcross\%$ genomes in $G(gen)$ based on $rank$ to mate
11:        replace $pcross\%$ genomes in $G(gen)$ with new offspring
12:        $gen \leftarrow gen + 1$
13:        mutate $pmut\%$ of genomes in $G(gen)$
14:        if $nconvergence \geq gen$ then
15:            if algorithm has converged then
16:                $converged \leftarrow$ TRUE
17:            end if
18:        end if
19:    end while
20:    return best genome
21: end procedure

22: procedure GETSCORE($ml, p_2, p_3, d_2, d_3, d_{sl}, ra$)
23:     run general selfish-mine simulation with given parameters
24:     return $R_p$ or $N_s$
25: end procedure

26: procedure OBJECTIVEFUNCTION($g$)
27:     $val \leftarrow$ GETSCORE($g(ml), g(p_2), g(p_3), g(d_2), g(d_3), g(d_{sl}), g(ra)$)
28:     return $val - benchmarkScore$
29: end procedure
```

The results of the first set of experiments in which we attempted to maximize R_p are shown in Tables 6.1 and 6.2. Tables 6.3 and 6.4 on the other hand show the results of the GA optimization when attempting to maximize N_s. In both cases, experiments were conducted based on two 40-node network models, the first model having instantaneous communication and the second, characterized by an average of 10 seconds communication delay. In addition, the distributed implementation of the general selfish-mine algorithm (no gatekeeper) was used while mining 5000 blocks (35 days of mining) for all simulation experiments conducted during the fitness evaluation of each in-

Table 6.1: GA optimization of R_p in a network with instantaneous block transmission.

α	N_s			R_p			Configuration (general selfish-mine)						
	H	S	G	H	S	G	ml	p_2	p_3	d_2	d_3	d_{sl}	ra
0.05	255	46	253	0.05	0.01	0.05	2	3	1	645	527	169	0
0.10	505	165	495	0.10	0.04	0.10	2	3	1	645	527	169	0
0.15	763	344	763	0.15	0.08	0.16	13	2	1	805	85	1186	0
0.20	1005	547	1004	0.20	0.13	0.21	5	2	1	461	908	927	0
0.25	1255	799	1255	0.25	0.19	0.27	13	2	1	805	85	1186	0
0.30	1493	1061	1493	0.30	0.27	0.33	13	2	1	805	85	1186	0
0.35	1743	1357	1738	0.35	0.36	0.40	14	1	1	1176	94	584	0
0.40	1995	1665	1966	0.40	0.45	0.50	14	1	1	1176	94	584	0
0.45	2243	1987	2063	0.45	0.56	0.63	14	1	1	1176	94	584	0
0.475	2365	2140	2365	0.47	0.61	0.73	14	1	1	1176	94	584	0

Table 6.2: GA optimization of R_p in a network with block transmission delays.

α	N_s			R_p			Configuration (general selfish-mine)						
	H	S	G	H	S	G	ml	p_2	p_3	d_2	d_3	d_{sl}	ra
0.05	252	79	252	0.05	0.02	0.05	2	3	1	645	527	169	0
0.10	496	223	494	0.10	0.05	0.10	2	3	1	645	527	169	0
0.15	753	442	761	0.15	0.10	0.16	13	2	1	805	85	1186	0
0.20	994	673	986	0.20	0.16	0.21	12	2	3	19	560	372	0
0.25	1242	957	1251	0.25	0.24	0.27	7	2	1	998	292	33	0
0.30	1476	1216	1477	0.30	0.31	0.33	5	2	1	461	908	927	0
0.35	1724	1493	1740	0.35	0.39	0.41	14	1	1	1176	94	584	0
0.40	1974	1795	1988	0.40	0.49	0.51	14	1	1	1176	94	584	0
0.45	2220	2085	2136	0.45	0.60	0.65	15	3	1	489	19	692	1
0.475	2342	2223	2293	0.47	0.70	0.73	15	3	1	489	19	692	1

Table 6.3: GA optimization of N_s in a network with instantaneous block transmission.

α	N_s			R_p			Configuration (general selfish-mine)						
	H	S	G	H	S	G	ml	p_2	p_3	d_2	d_3	d_{sl}	ra
0.05	255	46	253	0.05	0.01	0.05	2	3	1	645	527	169	0
0.10	505	165	505	0.10	0.04	0.10	13	2	1	805	85	1186	0
0.15	763	344	763	0.15	0.08	0.16	13	2	1	805	85	1186	0
0.20	1005	547	1003	0.20	0.13	0.21	7	2	2	348	24	160	0
0.25	1255	799	1255	0.25	0.20	0.27	13	2	1	805	85	1186	0
0.30	1493	1061	1493	0.30	0.27	0.33	13	2	1	805	85	1186	0
0.35	1743	1357	1738	0.35	0.36	0.41	13	2	1	805	85	1186	0
0.40	1995	1665	1995	0.40	0.45	0.50	9	2	1	1087	282	108	0
0.45	2243	1987	2224	0.45	0.56	0.62	9	2	1	1087	282	108	0
0.475	2365	2140	2365	0.50	0.61	0.66	7	2	1	998	292	33	0

dividual in a generation. We also note that the GA was able to find several equally ranked individuals (with different configurations of v), with equal fit-

Table 6.4: GA optimization of N_s in a network with block transmission delays.

α	N_s			R_p			Configuration (general selfish-mine)						
	H	S	G	H	S	G	ml	p_2	p_3	d_2	d_3	d_{sl}	ra
0.05	252	79	252	0.05	0.02	0.05	2	3	1	645	527	169	0
0.10	496	223	503	0.10	0.05	0.10	7	2	3	979	334	52	0
0.15	753	442	761	0.15	0.10	0.16	13	2	1	805	85	1186	0
0.20	994	673	1001	0.20	0.16	0.21	13	2	1	805	85	1186	0
0.25	1242	957	1248	0.25	0.24	0.27	13	2	1	805	85	1186	0
0.30	1476	1216	1480	0.30	0.31	0.32	3	2	1	504	132	99	0
0.35	1724	1493	1731	0.35	0.39	0.38	5	2	3	1092	42	739	0
0.40	1974	1795	1980	0.40	0.49	0.44	5	3	3	781	19	1158	0
0.45	2220	2085	2224	0.45	0.60	0.51	6	1	3	979	33	61	0
0.475	2342	2223	2342	0.47	0.70	0.55	4	2	2	955	0	1181	0

ness scores. In the tables, we only list the ones the GA selected randomly from a set of equal ranking best individuals at the end of each search for a given α. The column H gives the performance of the pool when following honest mining, column S, when following the standard selfish-mine strategy, and column G, when following the general selfish-mine strategy with the parameter vector v configured as shown is the columns labelled configuration (general selfish-mine). Based on the results in Table 6.1 to 6.4, we make the following observations.

- Maximizing R_p does not always result in a larger N_s value.
- Standard selfish-mine always performs worse than honest mining in terms of N_s even when R_p when the pool follows selfish-mine (column S) is greater than R_p when the pool follows honest mining (column H). This further confirms our findings in Chapter 5[3].
- There are configurations of the parameter vector v that yield higher R_p than standard selfish-mine [32] at the same time increasing the value of N_s, such that $N_s(selfish) \geq N_s(honest)$ for all α.
- N_s when $delay = 0$, is greater than N_s when $delay > 0$ ($delay$ as defined by equation 5.11 in Chapter 5 Section 5.2.3.3, taking $k = 1$).
- Even when the pool is small ($\alpha \approx 0.05$), implementing a well configured general selfish-mine strategy, allows the pool to perform slightly better than or at least the same as it would have by following honest mining both in terms of R_p and N_s.

[3]The results agree with those for standard selfish-mine with constant difficulty. Since only 5000 blocks are mined during the fitness evaluation of each individual, adjusting the mining difficulty has a negligible effect on the performance of the pool in terms on N_s as opposed to the results shown in Chapter 5 Section 5.2.3.2.

Exploiting the additional degrees of freedom that the parameters ml, p_2, p_3, d_2, d_3, d_{sl}, and ra afford the selfish pool, the pool can accrue higher revenue. For example, by choosing whether to engage in races or not when $lead = 0$, the pool can avoid wasting computational effort mining on the secret branch.

This can be beneficial in a network characterized by block transmission delays (Tables 6.2 and 6.4) when the pool contains a small number of nodes making the probability of winning the race minimal. Since our network has only 40 nodes, $\gamma < 0.5$ as shown in Figure 6.1 even when $\alpha = 0.5$ (only 20 nodes are dishonest). This means that the probability of the secret extension being extended in the next round of mining during a race is less than 0.5 unless the next block is mined by the pool. As a result, the pool may opt out of engaging in races as this may be more profitable as shown in Tables 6.1 to 6.4 ($ra = 0$ for all α). This is compounded by the possibility that due to block propagation delays, the pool may not be aware of a public block that has already been mined on top of the public branch but delayed to reach the pool.

In addition, for the case where there is instantaneous communication (Tables 6.1 and 6.3), the pool may opt out of engaging in races regardless of the number of nodes in the pool. This is because by the time a competing public block reaches the selfish pool, it will have reached all honest nodes as well, making it impossible for the secret extension to be extended in the next round of mining (taking into account the processing time required before the pool publishes the competing secret block) unless the next block is mined by the pool. However, if the pool has total network superiority, the pool can safely engage in races at all times.

We further selected the configuration of v found to be optimal for $\alpha = 0.4$ and ran some simulation experiments while varying the network superiority of the pool as done in Chapter 5 Section 5.2.3.3. The mining efficiency of the pool (P_s) when following honest mining, the standard selfish-mine strategy and the general selfish-mine strategy with $v = (5, 3, 3, 781, 19, 1158, 0)$ are plotted in Figure 6.2. We observe that the general selfish-mine strategy yields almost the same performance as honest mining when the pool is highly connected ($k = 0$ implying that $ns = 1$) but consistently slightly better than honest mining or much better than the standard selfish-mine strategy for the rest of the scenarios.

6.3 Conclusion

Notwithstanding the differences in approach, like [4] and [63], we conclude that there are more profitable block discarding strategies or more accurately, variations of the selfish-mine strategy as proposed in [32] that enable the pool to perform better than is possible with standard selfish-mine or st_1 [4]. These strategies enable the pool to accumulate relative revenue R_p in excess of its

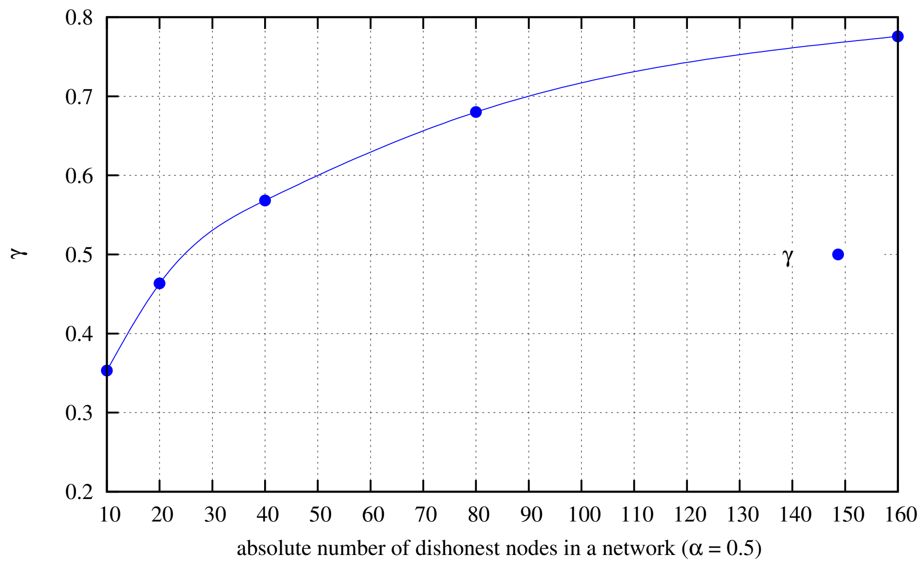


Figure 6.1: γ when $\alpha = 0.5$ for networks of varying sizes with an average communication delay of 10 seconds.

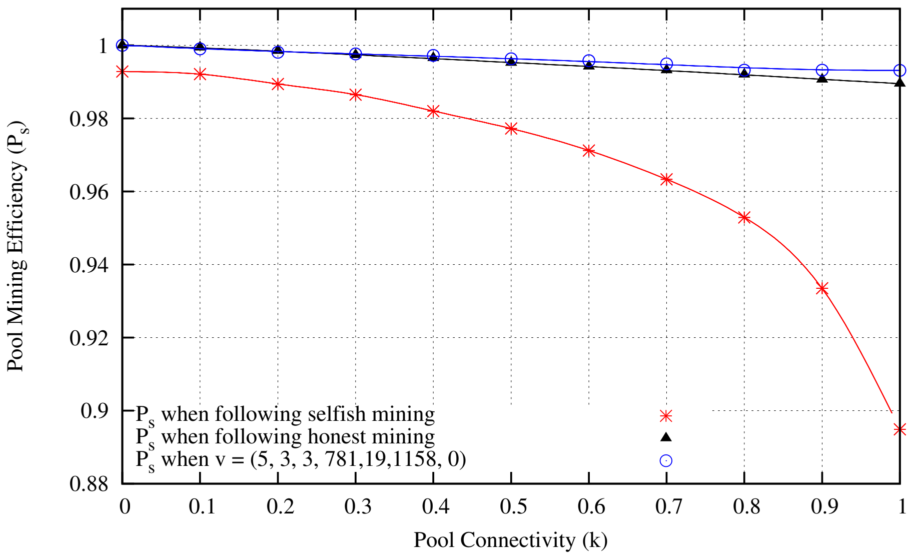


Figure 6.2: The performance of the pool (P_s) with varying k for $\alpha = 0.4$.

fair share even for small pool sizes ($\alpha \approx 0.15$), and the pool incorporates a statistically insignificant number of additional blocks (N_s) into the main chain than it would have by following honest mining.

Chapter 7

Summary and Conclusions

7.1 Summary

In this thesis, we investigated the performance of selfish bitcoin mining under block propagation delays. Chapter 1 presented a survey of related work, the background to the problem and research objectives. In Chapter 2, we presented an outline of the evolution of currencies and payments systems from the primitive barter economy to the present day of digital currencies and payment systems. We made a comparison of the various electronic payment systems and discussed in brief some of the popular altcoins. We endeavored to discuss the details of Bitcoin in Chapter 3. It was in this chapter where the architecture and underlying technical details of Bitcoin were presented, including an overview of the Bitcoin ecosystem. We further compared Bitcoin with other payment systems like Visa.

Beginning with Chapter 4 in which we introduced our discrete event simulator, we began presenting the results of our simulation of the Bitcoin blockchain dynamics. We first assumed in Chapter 4 that all nodes are honest, abiding by the Bitcoin protocol rules. We performed simulation experiments to validate our simulation model. We showed that our results agree with those observed from the real Bitcoin network [26], confirming that our simulations for the network model under investigation yield results that are close to those observed in the real Bitcoin network.

In Chapter 5, we assumed that a fraction of the miners commanding α of the total computing power of the network was dishonest. We presented the selfish-mine algorithm as proposed by [32] and defined our evaluation criteria, relative revenue, and mining efficiency. We presented the results based on the evaluation criteria for the various simulation experiments we conducted which were based on the simulation model introduced in Chapter 4, taking block propagation delays into account. In other experiments, we explicitly assumed instantaneous block transmission in the network and compared the results using the said evaluation criteria. We further investigated whether selfish mining is profitable and conducted some experiments to investigate the effectiveness of the Sybil attack proposed in [32] using a simple model. Finally, we showed that it is in the interest of the pool to follow honest mining.

In Chapter 6, we introduced the generalized selfish-mine algorithm based on the parameter vector $v = (ml, p_2, p_3, d_2, d_3, d_{sl}, ra)$. We performed a search for the optimal configuration of v that yields better performance for the pool than standard selfish mining using GA optimization. We showed that the optimal configuration of the general selfish-mine strategy as found by the GA performed consistently better than the standard selfish-mine strategy under all circumstances both in terms of the total blocks mined by the pool that end up in the main chain at the end of the simulation and relative revenue of the pool. We further showed, using the same criteria, that the general selfish-mine strategy performed as good as, but not better than honest mining.

7.2 Conclusions and Future Work

The goals of subversive mining strategies in the Bitcoin network seem to be twofold, namely,

1. to allow an attacker to gain more rewards than they would gain from following the protocol rules and
2. to break the Bitcoin protocol (explained below).

In view of this, we conclude that selfish mining is a threat to the Bitcoin system. In terms of the goals of subversive strategies presented above and when communication is instantaneous, selfish mining achieves both. Selfish mining as proposed in [32] not only allows the selfish pool to earn more than their fair share of the mining revenue (relative pool revenue), it potentially breaks the Bitcoin system.

Central to Bitcoin, is the idea of a decentralized payment system. For this reason, *breaking* Bitcoin should be understood in the following context. If selfish mining is successfully carried out by an attacker, the honest nodes perform poorly in terms of the revenue earned. Eventually, the honest nodes will either join the pool or leave the network, at which point Bitcoin ceases to be decentralized (*broken*). However, we also note that under block propagation delays, selfish mining as proposed in [32] only begins to yield profits, not in terms of the relative revenue but in terms of the actual blocks credited, when the pool commands more than 40% of the total hashing power of the network, provided the difficulty is adjusted downwards. Selfish mining is unprofitable in terms of actual blocks credited if the difficulty is kept constant. Notwithstanding this, if the pool can achieve total network superiority, selfish mining becomes profitable even when the pool is relatively small. Furthermore, we conclude that the best possible strategy an attacker can implement is an optimally configured general selfish-mine strategy. This increases the attacker's relative revenue, but the attacker accrues a statistically insignificant number of additional blocks in the main chain than the attacker would have when following honest mining. In essence, a well configured general selfish-mine strategy allows the pool to perform as well as when mining honestly for any pool size even under block propagation delays. In this sense, the generalized form of selfish mining poses a greater threat to Bitcoin than the standard selfish mining strategy.

However, the network model employed in our investigations was simplistic. It would be interesting to investigate these attacks using a more realistic network model that is as close to reality as possible, taking into account the Bitcoin P2P overlay network characteristics and topology. It would be interesting to learn whether this would have any effect on our findings and therefore presents an opportunity for possible future work. In addition, it would be insightful to carry out a thorough analysis of possible and already suggested measures to mitigate these block discarding attacks.

Appendices

Appendix A

Summarized Bitcoin Rules

The main branch of the blockchain is defined as the branch with highest total difficulty. The total difficulty of a branch is the time in expectation it took to mine the blocks in the branch. There are 3 categories of blocks,

1. blocks in the main branch; the transactions in these blocks are considered to be tentatively confirmed,
2. blocks on side branches off the main branch; these blocks have tentatively lost the race to be in the main branch,
3. orphan blocks; these are blocks which do not link into the main branch, normally because of a missing predecessor or n^{th}-level predecessor.

Blocks in the first two categories form a tree rooted at the genesis block, linked by the previous block pointers, which point towards the root. The tree is almost linear with a few short branches off the main branch.

A (tagged) block arrives or is mined at a node. The tagged block is either local (mined at this node) or foreign (mined at another node). Local blocks are added to the blockchain and broadcast to all the peers. As for the foreign block category, the following checks are done.

1. Reject the tagged block if a duplicate of the block is present in any of the three block categories.
2. Check if the previous block (matching the previous block hash in the tagged block) is in the main branch or a side branch. If not, add the tagged block to the orphan blocks and query the peer that sent the tagged block for the previous block.
3. If the previous block is in the main branch or a side branch, add the tagged block to the block chain. There are three cases.

    a) The tagged block extends the main branch: add the tagged block to the main branch.

b) The tagged block extends a side branch but does not add enough difficulty to make it become the new main branch: add the tagged block to the side branch.

c) The tagged block extends a side branch which becomes the new main branch: add the tagged block to the side branch

   i. find the fork block on the main branch from which this side branch forks off,

   ii. redefine the main branch to only go up to the fork block,

   iii. for each block on the side branch, from the child of the fork block to the leaf, add the block to the main branch,

   iv. delete each block in the old main branch, from the leaf to the child of the fork block.

d) For each orphan block, check if the previous block now points to a valid block in either the main chain or side branch. If so, add the block to the blockchain and perform checks in (c), or else, leave the block as is.

Appendix B

The Three Node Network

B.1 The Three Node Network

The performance of the selfish-mine strategy to some extent depends on the fraction γ of honest miners who mine on the published block when there is a race. According to Bitcoin rules, a miner will mine on the first block received. Unlike in [32], where γ is manipulated as wished, in our case, γ is an observation that is measured.

In order to analyze γ, a three node network configuration was studied. Three nodes A, B and C are placed on the (x, y) plane as shown in Figure B.1. Nodes A and B are honest while node C is dishonest. We assume the communication delay between two nodes is proportional to the distance between the nodes. In this case, it takes ka seconds to send a message from A to B, kb seconds from B to C and kc seconds from A to C, where k is a constant. We consider two approaches; an analytic approach and a simulation model. The results of our simulations agree with the analytic results.

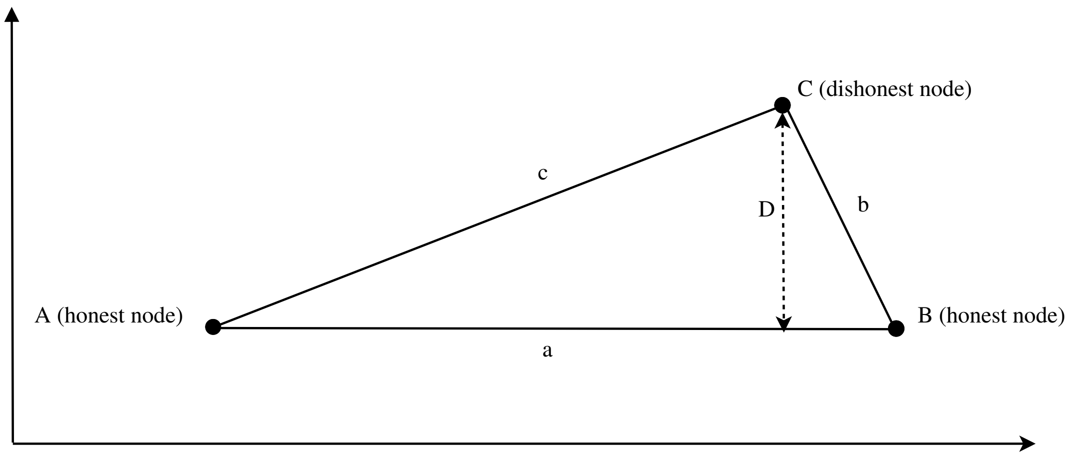


Figure B.1: The three node network configuration.

B.2 Analytic Model

Consider the situation where an honest node A discovers a block A_h and transmits it to all nodes[1]. When the block arrives at the dishonest node C, this node already knows of a secret block C_s that comprises a secret extension of length one. Consequently, the blockchain at C forks and the dishonest node C immediately publishes the secret block C_s causing a race to begin at node C and, after a delay, at nodes A and B. The race at node A is won by block A_h. The outcome of the race at node B is more complicated.

Assume that the transmission times between nodes i and j are normally distributed with a mean proportional to the distance between the two nodes and with a constant coefficient of variation CV. We are interested in the probability P that the honest node B receives the block C_s before it receives block A_h. In the notation of [32], P is referred to as γ.

Let T be the difference between the arrival time of C_s and A_h at B. Then,

$$T = T_{AC} + T_{CB} - T_{AB}, \tag{B.1}$$

where T_{ij} is normally distributed with mean kd_{ij} and variance σ^2. Therefore T is normally distributed with mean

$$t = k(b + c - a) \tag{B.2}$$

and variance

$$\sigma^2 = k^2(CV)^2(a^2 + b^2 + c^2), \tag{B.3}$$

where CV is the coefficient of variation. We wish to calculate the probability P that $T \leq 0$.

A, B and C form a triangle of height D as shown in Figure B.1. Using the *cosine law*, we have,

$$b^2 = a^2 + c^2 - 2ac\cos\theta, \tag{B.4}$$

where θ is the angle between a and c. To compute the value of $\cos\theta$, we first calculate

$$\sin\theta = D/c. \tag{B.5}$$

Using the identity,

$$\sin^2\theta + \cos^2\theta = 1 \tag{B.6}$$

and equation B.5, we obtain,

$$\cos\theta = \sqrt{1 - \sin^2\theta} = \sqrt{1 - (D/c)^2}, \tag{B.7}$$

thus

$$b^2 = a^2 + c^2 - 2ac\sqrt{1 - (D/c)^2}. \tag{B.8}$$

[1]The model presented in this section is attributed to P.G. Taylor (private communication).

When $D = 0$, then A, B and C are collinear and $P = 0.5$.

We now compute the probability P that $T \leq 0$ for values of $0 < D \leq 100$, $a = 800$, $c = 500$ and $CV \in \{0.001, 0.002, 0.005, 0.01\}$. In order to compute the probability P, we use the Python program given in Appendix C.

The results are shown in Table B.1.

Table B.1: Probability P with varying D.

$CV = 0.001$		$CV = 0.002$		$CV = 0.005$		$CV = 0.01$	
D	P	D	P	D	P	D	P
1.0	0.4989	1.0	0.4995	1.0	0.4998	1.0	0.4999
10.0	0.3939	10.0	0.4465	10.0	0.4785	10.0	0.4893
20.0	0.141	20.0	0.2953	20.0	0.4148	20.0	0.4572
30.0	0.007865	30.0	0.1136	30.0	0.3145	30.0	0.4046
40.0	9.303E-06	40.0	0.01616	40.0	0.1959	40.0	0.3343
50.0	1.332E-11	50.0	0.0004311	50.0	0.0913	50.0	0.2526
60.0	6.298E-22	60.0	8.918E-07	60.0	0.02803	60.0	0.1697
70.0	1.437E-38	70.0	4.99E-11	70.0	0.004842	70.0	0.09793
80.0	1.363E-63	80.0	2.296E-17	80.0	0.0003916	80.0	0.04654
90.0	2.997E-99	90.0	2.365E-26	90.0	1.207E-05	90.0	0.01737
100.0	6.082E-148	100.0	1.367E-38	100.0	1.137E-07	100.0	0.004831

B.3 Simulation Model

We performed simulation experiments for the three node network configuration shown in Figure B.1 while varying D from 0 to 100 and values of the coefficient of variation $CV \in \{0.001, 0.002, 0.005, 0.01\}$. The communication delay between two nodes has standard deviation,

$$\sigma = kdCV \tag{B.9}$$

where d is the distance between the two nodes. The results are presented in Figure B.2. The graph shows that $\gamma = 0.25$ when $D = 0$. This is due to how γ is calculated in the simulator. Consider the case where node A mines all public blocks and broadcasts them to nodes B and C. When a block reaches dishonest node C, it responds by publishing its secret block. A race-counter is incremented and a γ-counter is incremented or not depending on which block arrives first at B. However, since A discovered the block, the secret block from C can never reach A before the public block hence the γ-counter is never incremented at A. At the end of the simulation, the γ-counter/race-counter is divided by the number of honest nodes, in this case 2, effectively halving the measurement. However, when the network size increases, this effect becomes negligible. A normalized form of the graph that takes the above into consideration is shown in Figure B.3 showing that the observed value of $\gamma = 0.5$ in the simulation model is equal to the probability $P = 0.5$

computed in the analytic model in the three node network when D is zero (the nodes are collinear). In addition, the (D,P) pairs in Table B.1 and the (D,γ) pairs in Figure B.3 for corresponding values of CV, closely agree.

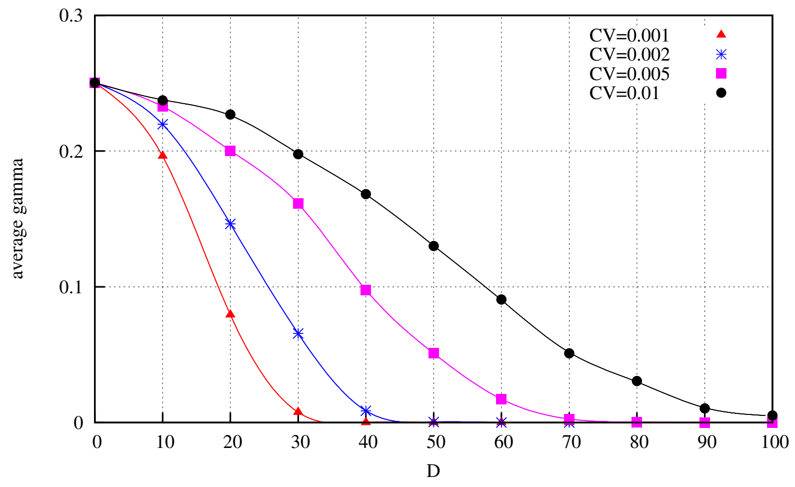


Figure B.2: γ for different values of CV in a 3 node network.

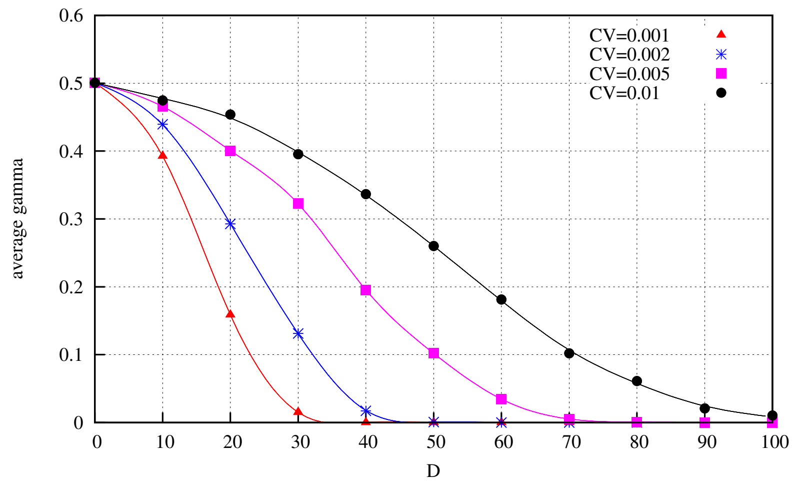


Figure B.3: Normalized γ for different values of CV in a 3 node network.

Appendix C

Numerical Program to Calculate P in Python

```python
import scipy.stats as ss
import math as math

 #values of D
d_list=[1.0,10.0,20.0,30.0,40.0,50.0,60.0,70.0,80.0,90.0,100.0]

#values of CV
cv_list=[0.01, 0.005,0.002,0.001]

#compute the mean given that a,b,c are normally distributed
def compute_mean(a,b,c):
  return b+c-a

#compute the standard deviation given that a,b,c are normally
    distributed
def compute_std_deviation(a,b,c,CV):
  return math.sqrt(CV*CV*(a*a + b*b +c*c))

#compute the probability P that P <= x given the mean and standard
    deviation
def compute_probability(x,mean,std_deviation):
  return ss.norm.cdf(x, loc = mean, scale=std_deviation)

#do the calculations for varying values of CV and D
def do_computations(a,c):
  for CV in cv_list:
    print "CV = ", CV
    for D in d_list:
      #compute the cosine of the angle opposite the unknown side b
      cosB = math.sqrt(1 - (D/c)*(D/c))
      b = math.sqrt(a*a + c*c - 2*a*c*cosB) #compute unknown triangle
          side b
      mean = compute_mean(a,b,c)
      std_deviation = compute_std_deviation(a,b,c,CV)
      print D,"%.4G" % (compute_probability(0,mean,std_deviation))
    print "\n"

#call to execute do_computations for a=500 and c=100
do_computations(800,500)
```

List of References

[1] Adams, G. (2012). *Letters to John Law*. Newton Page. ISBN 9781934619087.
Available at: https://books.google.co.za/books?id=espxkAw-5bsC

[2] Antonopoulos, A. (2014). *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. O'Reilly Media. ISBN 9781491921982.
Available at: https://books.google.co.za/books?id=k3qrBQAAQBAJ

[3] Back, A. (2002). Hashcash-a denial of service counter-measure.
Available at: http://www.hashcash.org/papers/hashcash.pdf

[4] Bahack, L. (2013). Theoretical bitcoin attacks with less than half of the computational power (draft). *arXiv preprint arXiv:1312.7013*.

[5] Bailey, G., Law, F., Phillips, M. and Brooks, R. (2006). *Cowries, Coins, Credit: The History of Money*. My Money Series. Compass Point Books. ISBN 9780756516765.
Available at: https://books.google.co.za/books?id=Lg5XCJ_rPdMC

[6] Barber, S., Boyen, X., Shi, E. and Uzun, E. (2012). Bitter to better - how to make bitcoin a better currency. In: *Financial Cryptography and Data Security*, pp. 399–414. Springer.

[7] Bernholz, P. (2015). *Monetary Regimes and Inflation: History, Economic and Political Relationships, Second Edition*. EBL-Schweitzer. Edward Elgar Publishing, Incorporated. ISBN 9781784717636.
Available at: https://books.google.co.za/books?id=u8TiBwAAQBAJ

[8] Bitcoin (2015). Download - bitcoin. [Online; accessed 12-July-2015].
Available at: https://bitcoin.org/en/download

[9] Bitcoin-Wallet (2015). Bitcoin wallet - android apps on google play. [Online; accessed 12-July-2015].
Available at:
https://play.google.com/store/apps/details?id=de.schildbach.wallet

[10] BitcoinWatch (2015). Bitcoin watch. [Online; accessed 23-June-2015].
Available at: http://www.bitcoinwatch.com/?title=BitcoinWatch

[11] BitPay (2015). Accept bitcoin | bitpay. [Online; accessed 12-July-2015].
Available at: https://bitpay.com/

[12] BitPesa (2015). Bitpesa. [Online; accessed 12-July-2015].
Available at: `https://www.bitpesa.co/`

[13] BitX (2015). Bitx. [Online; accessed 12-July-2015].
Available at: `https://bitx.co/`

[14] Blockchain.info (2015). Bitcoin wallet. [Online; accessed 22-April-2015].
Available at: `https://blockchain.info/wallet`

[15] Bossone, B. and Cirasino, M. (2001). *The Oversight of Payments Systems: A Framework for the Development and Governance of Payment Systems in Emerging Economies*. Payments and securities clearance and settlement systems research series. Centro de Estudios Monetarios Latinoamericanos. ISBN 9789686154764.
Available at: `https://books.google.co.za/books?id=JagZAAAACAAJ`

[16] Bradford, T. and Keeton, W.R. (2012). New person-to-person payment methods: have checks met their match?
Available at: `https://www.kansascityfed.org/publicat/econrev/pdf/12q3Bradford-Keeton.pdf`

[17] CiPHRAX (2015). msigna by ciphrax. [Online; accessed 12-July-2015].
Available at: `https://ciphrex.com/products/`

[18] Circle (2015). Circle internet financial. [Online; accessed 22-April-2015].
Available at: `https://www.circle.com/en/about`

[19] Coinbase (2015). Bitcoin wallet. [Online; accessed 22-April-2015].
Available at: `https://www.coinbase.com/`

[20] CoinDesk (2015). How kipochi is taking bitcoin into africa. [Online; accessed 12-July-2015].
Available at: `http://www.coindesk.com/kipochi-taking-bitcoin-africa/`

[21] Copay (2015). Copay - secure, shared bitcoin wallet. [Online; accessed 12-July-2015].
Available at: `https://copay.io/`

[22] Courtois, N.T. (2014). On the longest chain rule and programmed self-destruction of crypto currencies. *arXiv preprint arXiv:1405.0534*.

[23] Courtois, N.T. and Bahack, L. (2014). On subversive miner strategies and block withholding attack in bitcoin digital currency. *arXiv preprint arXiv:1402.1718*.

[24] Dalal, N. (2014). Exploring the bitcoin system: a complex econo-sociotechnical systems (cest) perspective. *International Journal of Conceptions on Management and Social Sciences*, vol. 2, pp. 47–51.

[25] Davies, G. (2010). *History of Money.* University of Wales Press. ISBN 9781783162765.
Available at: `https://books.google.co.za/books?id=Ym2uBwAAQBAJ`

[26] Decker, C. and Wattenhofer, R. (2013). Information propagation in the Bitcoin network. In: *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pp. 1–10. IEEE.

[27] Duffield, E. and Hagen, K. (2014). Darkcoin: Peer-to-peer currency with anonymous blockchain transactions and an improved proof-of-work system.
Available at:
`https://www.darkcoin.io/downloads/DarkcoinWhitepaper.pdf`

[28] Ebrey, P. and Walthall, A. (2013). *Pre-Modern East Asia: A Cultural, Social, and Political History, Volume I: To 1800.* Cengage Learning. ISBN 9781285546230.
Available at: `https://books.google.co.za/books?id=BZ8WAAAAQBAJ`

[29] Econet (2015). Ecocash Zimbabwe. [Online; accessed 22-April-2015].
Available at: `https://www.econet.co.zw/ecocash/`

[30] European-Central-Bank (2012). Virtual currency schemes.
Available at: `https://www.ecb.europa.eu/pub/pdf/other/virtualcurrencyschemes201210en.pdf`

[31] Eyal, I. (2014). The miner's dilemma. *arXiv preprint arXiv:1411.7099.*

[32] Eyal, I. and Sirer, E.G. (2014). Majority is not enough: Bitcoin mining is vulnerable. In: *Financial Cryptography and Data Security*, pp. 436–454. Springer.

[33] Franco, P. (2014). *Understanding Bitcoin: Cryptography, Engineering and Economics.* John Wiley & Sons.
Available at: `https://books.google.co.za/books?id=erMQBQAAQBAJ&lr=&source=gbs_navlinks_s`

[34] Göbel, J., Groenewald, G., Krzesinski, A. and Mwale, M. (2015). Simulation modelling of bitcoin blockchain dynamics. In: *Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2015*, pp. 201–206. SATNAC.

[35] Göbel, J., Keeler, P., Krzesinski, A.E. and Taylor, P.G. (2015). Bitcoin blockchain dynamics: the selfish-mine strategy in the presence of propagation delay. *arXiv preprint arXiv:1505.05343.*

[36] Göbel, J. and Krzesinski, A. (2015). Bitcoin blockchain dynamics and dishonest mining (in preparation).

[37] Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Artificial Intelligence. Addison-Wesley Publishing Company, Boston, MA, USA. ISBN 9780201157673.
Available at: `https://books.google.co.za/books?id=3_RQAAAAMAAJ`

[38] Halaburda, H. and Gandal, N. (2014). Competition in the cryptocurrency market. *Social Science Research Network (SSRN)*.
Available at:
`http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2506463`

[39] Headrick, D.R. (2009). *Technology: A World History*. New Oxford World History. Oxford University Press, USA. ISBN 9780199713660.
Available at: `https://books.google.co.za/books?id=qG2tPzkN6HUC`

[40] Heilman, E. (2014). One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner. *IACR Cryptology ePrint Archive*, vol. 2014, p. 7.

[41] Herald, S.M. (2015). The rise of dogecoin, the internets hottest cryptocurrency. [Online; accessed 22-April-2015].
Available at: `http://www.smh.com.au/technology/technology-news/the-rise-and-rise-of-dogecoin-the-internets-hottest-cryptocurrency-20140124-31d24.html`

[42] Huang, D.Y. (2013). Profit-driven abuses of virtual currencies. *University of California, San Diego*.
Available at: `http://sysnet.ucsd.edu/~dhuang/pmwiki/uploads/Main/huang-research-exam.pdf`

[43] Johnson, B., Laszka, A., Grossklags, J., Vasek, M. and Moore, T. (2014). Game-theoretic analysis of ddos attacks against bitcoin mining pools. In: *Financial Cryptography and Data Security*, pp. 72–86. Springer.

[44] Karame, G.O., Androulaki, E. and Capkun, S. (2012). Two bitcoins at the price of one? Double-spending attacks on fast payments in bitcoin. *IACR Cryptology ePrint Archive*. `http://eprint.iacr.org/`.

[45] King, S. (2013). Primecoin: Cryptocurrency with prime number proof-of-work.
Available at: `http://primecoin.io/bin/primecoin-paper.pdf`

[46] King, S. and Nadal, S. (2012). Ppcoin: Peer-to-peer crypto-currency with proof-of-stake.
Available at:
`http://wallet.peercoin.net/assets/paper/peercoin-paper.pdf`

[47] Lee, Z.-Y., Yu, H.-C. and Ku, P.-J. (2001). An analysis and comparison of different types of electronic payment systems. In: *Management of Engineering and Technology, 2001. PICMET'01. Portland International Conference on*, pp. 38–45. IEEE.

[48] Mankiw, G.N. (2014). *Principles of Economics*. Seventh edn. Cengage Learning, Stamford, CT, USA. ISBN 9781285165875.

[49] MediaWiki (2015). Protocol rules - bitcoin. [Online; accessed 8-April-2015]. Available at: `https://en.bitcoin.it/wiki/Protocol_rules?title=ProtocolRules-Bitcoin`

[50] Merkle, R.C. (1988). A digital signature based on a conventional encryption function. In: *Advances in Cryptology-CRYPTO'87*, pp. 369–378. Springer.

[51] Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. A Bradford book. MIT Press, Cambridge, MA, USA. ISBN 9780262631853.
Available at: `https://books.google.co.za/books?id=0eznlz0TF-IC`

[52] MoneyBookers (2015). Moneybookers. [Online; accessed 16-October-2015].
Available at: `http://www.howtopayonline.org/moneybookers.html`

[53] MTN (2015). Mobile money. [Online; accessed 22-April-2015].
Available at: `https://www.mtn.co.za/everyday_services/useful_extras/Banking/Pages/mobile_banking.aspx`

[54] Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system.
Available at: `https://bitcoin.org/en/bitcoin-paper`

[55] Nano, L. (2015). Ledger wallet - smartcard based bitcoin hardware wallet. [Online; accessed 12-July-2015].
Available at: `https://www.ledgerwallet.com/`

[56] Naor, M. and Dwork, C. (1993). Pricing via processing, or, combatting junk mail, advances in cryptology. In: *CRYPTO'92: Lecture Notes in Computer Science No. 740*, pp. 139–147. Springer.
Available at: `www.wisdom.weizmann.ac.il/~naor/PAPERS/pvp.ps.gz`

[57] Nayak, K., Kumar, S., Miller, A. and Shi, E. (2015). Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. Cryptology ePrint Archive, Report 2015/796. `http://eprint.iacr.org/`.

[58] Okupski, K. (2014). Bitcoin protocol specification.
Available at: `http://www.enetium.com/resources/Bitcoin.pdf`

[59] PayFast (2015). Accept bitcoin payments with payfast. [Online; accessed 12-July-2015].
Available at: `https://www.payfast.co.za/2014/07/17/accept-bitcoin-payments-payfast/`

[60] PayPal (2015). Send money, pay online or set up a merchant account - paypal. [Online; accessed 08-July-2015].
Available at: `https://www.paypal.com/za/webapps/mpp/home`

[61] PCWorld (2015). Paypal will let you pay with Bitcoin, sometimes | PC World. [Online; accessed 09-July-2015].
Available at: `http://www.pcworld.com/article/2687456/paypal-will-let-you-pay-with-bitcoin-sometimes.html`

[62] Peeters, K. (2015). tree.hh:an STL-like C++ tree class. [Online; accessed 16-October-2015].
Available at: `http://tree.phi-sci.com/`

[63] Sapirshtein, A., Sompolinsky, Y. and Zohar, A. (2015). Optimal selfish mining strategies in bitcoin. *arXiv preprint arXiv:1507.06183*.

[64] Saylor, M. (2012). *The Mobile Wave: How Mobile Intelligence Will Change Everything*. Vanguard Press, Boston, MA, USA. ISBN 9781593157203.
Available at: `https://books.google.co.za/books?id=X9Gmf98YdgIC`

[65] Shomer, A. (2014). On the phase space of block-hiding strategies in bitcoin-like networks. *arXiv preprint arXiv:1402.4233*.

[66] Snapcard (2015). Accept bitcoin payments. [Online; accessed 22-April-2015].
Available at: `https://www.snapcard.io/`

[67] Summers, B. (2012). *Payment systems : design, governance and oversight*. Central Banking Publ. ISBN 9781902182728.
Available at: `https://books.google.co.za/books?id=y9jAlgEACAAJ`

[68] SWIFT (2015). Swift - the global provider of secure financial massaging services. [Online; accessed 23-June-2015].
Available at: `http://www.swift.com/index.page?lang=en`

[69] TechEU (2015). A guide to bitcoin (part ii): A deep delve into the bitcoin ecosystem. [Online; accessed 09-July-2015].
Available at: `http://tech.eu/features/926/bitcoin-ecosystem/`

[70] TheEconomist (2015). Forking hell | The Economist. [Online; accessed 31-August-2015].
Available at: `http://www.economist.com/news/business-and-finance/21661404-spat-between-developers-may-split-digital-currency-forking-hell`

[71] Trezor (2015). Trezor: The bitcoin safe. [Online; accessed 12-July-2015].
Available at: `https://www.bitcointrezor.com/`

[72] Visa (2015). About Visa - Visa USA. [Online; accessed 08-July-2015].
Available at: `http://usa.visa.com/about-visa/index.jsp`

[73] Vodacom (2015). M-Pesa - SA's mobile financial services | Vodacom. [Online; accessed 24-April-2015].
Available at: `http://www.vodacom.co.za/vodacom/services/financial-solutions/m-pesa`

[74] Wikipedia (2015). Incentive compatibility — wikipedia, the free encyclopedia. [Online; accessed 13-May-2015].
Available at: `http://en.wikipedia.org/w/index.php?title=Incentive_compatibility&oldid=659798458`

[75] Wong, H. (1996). Introduction to genetic algorithms. [Online; accessed 22-April-2015].
Available at: `http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html`

[76] Zhao, Y. (2015). Cryptocurrency brings new battles into the currency market. In: *Proceedings of the Seminars Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)*, pp. 91–99. Technische Universität München, Germany. ISBN 978-3-937201-47-4.