TÉCNICO LISBOA

Web Monetization Using Crypto-Currencies

Natalino António Bernardino Cordeiro

Thesis to obtain the Master of Science Degree in

Information Systems and Software Engineering

Supervisor: Prof. Miguel Nuno Dias Alves Pupo Correia

Examination Committee

Chairperson: Prof. João António Madeiras Pereira
Supervisor: Prof. Miguel Nuno Dias Alves Pupo Correia
Member of the Committee: Prof. Pedro Miguel dos Santos Alves Madeira Adão

April 2019

Dedicated to my mother, father and sister.

For their support throughout this journey.

Acknowledgments

I would like to thank my advisor Miguel Pupo Correia, for giving the freedom to work according to my limitations and for his tireless availability.

I would also like to thank my friends Inês Percheiro and Luís Freixinho, for supporting me through one of the most challenging periods of my life.

Finally I want to thank my family, who worked so hard to give me the chance to be delivering this thesis. Without them I couldn't have done it.

Resumo

Hoje em dia, a maioria do conteúdo produzido para a internet, depende de modelos de receita baseados em publicidade, para pagar pelos serviços prestados. Com a popularização de extensões bloqueadoras de publicidade, as receitas obtidas através de publicidade têm diminuído drasticamente, obrigando websites a adotar medidas alternativas para monetizar o seu conteúdo. Serviços como o CoinHive, oferecem a opção de usar o poder computacional dos CPUs dos utilizadores dessas páginas para minar cripto-moedas, como forma alternativa de gerar lucro. Mas a baixa rentabilidade do serviço, taxas elevadas, flutuação da valoração das cripto-moedas, e consumo agressivo dos recursos dos utilizadores, dificultam a sua adoção por empresas estabelecidas no mercado. Este projeto visa oferecer uma prova de conceito de uma alternativa ao CoinHive e outros serviços de web-minining.

O projeto desenvolvido, consiste num serviço de web-minining, concebido para trabalhar com grupos de mineração de terceiros. O projeto visa oferecer a donos de websites a possibilidade de ter a seu próprio serviço de web-minining, em vez de dependerem de alternativas de terceiros, que cobram uma percentagem dos lucros como taxas de pagamento.

O objetivo final é entender o quão rentável este tipo de serviços pode ser e, com base nesses resultados, avaliar se eles irão proliferar no futuro ou cair em desuso, como já aconteceu no passado.

Keywords: Cripto-moedas, CoinHive, Extensões bloqueadoras de publicidade, Web-miner, Grupos de mineração, Modelo de receita

Abstract

Most web content produced today relies on ad-based revenue models to pay for their services. With the popularization of ad-blockers, ad revenue has been drastically reduced, forcing websites to adopt more alternative measures to monetize their content. Services like CoinHive, offer the option to use website visitors' unused CPU power to mine cryptocurrency as an alternative revenue source, but the low profitability of the service, high fees, fluctuation of cryptocurrency valuation and aggressive consumption of user resources, difficult its adoption by reputable businesses. This project aims to provide a proof-of-concept alternative to CoinHive and other web-mining services.

The developed project, consists in a web-mining service, designed work with third-party mining pools, with a emphasis on transparency and user control over their participation in the mining process. It aims to allow website owners to run their own web-mining service, instead of having rely on third-party alternatives which take a percentage of the mined rewards as payment fees.

The end goal is to gain an understanding of how much revenue these kind of services can generate and, based on those results, assess if they will proliferate in the future or fall into disuse, as they have in the past.

Keywords: Cryptocurrency, CoinHive, Ad-Blockers, Web-miner, Mining pools, Revenue model

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Since its creation, the Internet has spread so widely that today essentially every person on the planet knows what it is, and most have access to it. The Internet is present in every step of the daily routine of the modern society citizen, and consequently most individuals, organizations and businesses, have an online presence of some kind. It changed the way we communicate, shop, read the news, consume entertainment media, and alongside all of this, it created a detachment between consuming digital goods and paying for them.

Advertisements, including sponsored content, are the main source of income keeping web businesses that provide "free" content afloat. But with the increased adoption of ad-blocking browser extensions, partially in response of the obnoxious quantity of ads present on certain websites, this business model is becoming less viable. In turn, fake news, non-disclosed sponsored content and click baiting became more common trends, creating a toxic atmosphere throughout the internet. Other practices like collecting and selling user information to third parties without disclosure or explicit permission, are also common in the web and further add to the climate of user distrust in internet corporations.

With the exception of a few payed subscription-based services, the common user has the assumption that content comes for free on the internet. Crimes like stealing, which are condemned by society, became accepted under the alias of "internet downloads", mainly as consequence of the "everything is free" mentality. The phrase *If you are not paying for a product; You are not the customer; You are the product*, was popularized as a means to explain how consumer's data and behavior can generate a profit for businesses, and although it is not empirically correct, helps show that one way or the other if you are consuming something you are probably paying for it, even if you are not aware of that.

Money on its own has no more value than the one associated with the materials used to print it. However, money is a recognized way to represent and exchange credit between participants,

working as a placeholder for value. This mechanism, that fuels the world economy, despite being far from perfect, has been the reigning option since it was first invented many millennia ago. The need for recognition makes money heavily dependent on institutions and governments, instead of the population that uses it. Two examples of how the current system is flawed, are the Zimbabwe 100 trillion-dollar bill and the financial crisis of 2007/08.

1.1 Cryptocurrencies

It has been long since people first realized the shortcomings of physical money. Digital cash technologies, promising to solve many of the underlying issues associated with traditional cash, have been in the works since as early as David Chaum's eCash, conceived in 1983. This eCash idea, highly supported by adherents of the Cypherpunk movement, led to the creation of a large number of early digital cash projects, but ultimately all failed to gain any real traction. Curiously, just a few months after the start of the financial crisis of 2007/08, on 31 October 2008, the whitepaper *Bitcoin: A Peer-to-Peer Electronic Cash System* [1] by Satoshi Nakamoto, was released. The paper proposed a decentralized digital currency enabled by a peer-to-peer network, where transactions between users are done directly, without an intermediary, removing the need for a central bank or administrator. By using cryptography and nodes from the network, the transactions are verified and recorded in a public ledger called a blockchain. The process of verifying these transactions and guaranteeing the consensus of the ledger is called mining and is rewarded with the currency.

The growth and popularization of the *Bitcoin* project led to a huge increase in new cryptocurrency-related start-ups and in speculative interest from the public. Nowadays, whilst new projects seem to be announced every other day, an equal number of existent start-ups file for bankruptcy or turn out to be scams. Nonetheless, proposals like *Ethereum*, *Ripple*, *Litecoin* or *Monero*, stand out from the sea of cryptocurrencies currently being traded, due to their better development teams, liquidity or application of decentralized computing.

1.2 Topic Overview

Services like CoinHive [2] bring a twist to the usual process of mining, by having adherent entities inject JavaScript into their web pages, consuming CPU resources from the website readers to mine the cryptocurrency Monero. Adherents of the technology are then payed a percentage of the coins mined by their users, effectively monetizing the webpage in which the JavaScript code was injected and posing as an alternative to traditional ad-based monetization.

This practice, called *in-browser mining*, used by CoinHive could theoretically serve as a valid alternative source of revenue for web businesses, but there are a series of factors that prevent it, and similar services, from being a viable option for reputable companies:

- Most cryptocurrencies fail to have a concrete value proposition, meaning there is no actual real-world utility behind the service, and that their valuation is highly speculative and liable to crashing;
- The high fluctuation of cryptocurrency pricing makes it difficult to have an accurate revenue expectation;
- Blockchain based protocols currently have very poor scalability, affecting the long-term feasibility of such alternatives;
- In-browser mining works by injecting JavaScript into web pages, which enables all sorts of possibilities for hackers, namely man-in-the-middle attacks on open-networks [3][4][5], and diminishes the need for user input on whether or not to use the service
- There are no mechanisms in place to safeguard users' control over how much of their hardware and data resources are used. This can lead to users losing trust in a website and stopping to access it.

1.3 Objectives

This project aims to provide an understanding about how these new alternative channels of revenue for web businesses perform in terms of reliability and profitability, and how they can be improved.

The presented solution should provide an alternative to the existent services, such as CoinHive, whilst addressing some of the issues that pose as hindrances to the adoption of in-browser mining as an alternative to advertisements and other traditional web revenue models. The end goal is to find out, through a proof-of-concept, if a service of this nature can be sustainable today, and assess how it could develop in the future, in terms of adoption and revenue. To achieve those objectives the requirements are:

- The chosen or defined cryptocurrency to be mined, needs to have a real-world user-base;
- The system should grant website owners more flexibility and incur less fees than the competition;

- The system should enable users to decide whether or not to use the technology;
- The system should safeguard users, giving them control over their CPU power usage.

1.4 Thesis Outline

The remaining of this dissertation is structured as follows. Chapter 2 examines the state of the art works that served as bases for this thesis. Chapter 3 explains the design and preliminary concepts that were applied on the developed project. Chapter 4 covers the implementation of the developed project, stating its principal algorithms and their functionality. Chapter 5 presents the experimental evaluation of the developed solution and discusses the obtained results. Chapter 6 concludes the dissertation.

Chapter 2

Background and Related Work

This chapter presents and overview of the background and work related to the developed project. Section 2.1 introduces the most relevant revenue models used by web businesses. Section 2.2 introduces the concept of cryptocurrency. Section 2.3 explains in detail what the blockchain is and how it is essential for cryptocurrencies to exist. Section 2.4 introduces what mining pools are and how they work. Section 2.5 explains various algorithms used to achieve distributed consensus in cryptocurrencies. Section 2.6 details some of the problems that difficult the scalability of cryptocurrencies.

2.1 Revenue Models for Web Businesses

When the first Internet browser was invented in 1990 by Tim Berners-Lee, it opened a gateway for regular people to access online content for the first time. As a consequence, a new market where businesses could thrive, expand and become reachable worldwide with relative ease, was also created. Fast forward to today and having an online presence has become a requirement for any company's success. Certain businesses even exist exclusively on the Internet, having no physical stores or re-sellers connecting their services to their customers. Naturally, these web companies also need revenue models adapted to their limitations and often different from the ones used by traditional retail-focused corporations.

Advertisement based models, including CPM (Cost-Per-Mile), CPC (Cost-Per-Click), CPA (Cost-Per-Acquisition) and sponsorship deals, fuel most perceptually free content present in the Internet. Content creators, ranging from blogs to news websites and streaming services, often have both ad-based and subscription-based models, relying on ads or limited access to reach most of their audience, whilst giving users the option to pay a subscription fee in exchange for ad-free full access to their content. E.g. Spotify, YouTube and Wired. Besides advertisements

and subscription fees, other popular revenue streams include Pay-Per-View access, online retail, micro-transactions and usage of subscribers' data for marketing and research.

The rise of blockchain networks propelled a new approach to advertising on the web, called *Brave* [6]. This project, from the creator of JavaScript and co-founder of Mozilla, intends to create a decentralized ad exchange that removes middleman from the online advertising marketplace, and consequently helps to better monetize publisher content while protecting user privacy. It relies on Brave, an open source, privacy-focused browser that blocks third party ads and trackers, as its base and integrates the Basic Attention Token (BAT) [7] cryptocurrency as its payment system, that rewards and protects the user while giving better conversion to advertisers and higher yield to publishers. In the Brave ecosystem, when users see ads both publishers and users receive BAT, with most of the tokens going to the publishers. Users can then give the tokens they received back to publishers they wish to support. Additionally, publishers can also charge BAT for premium content or pay users with BAT when they promote their content.

CoinHive

With the expansion in the cryptocurrencies field, a new way to generate revenue was invented, *in-browser mining*. When *Bitcoin* became gradually harder to mine, due to the increased usage of GPUs and ASIC miners, this practice was abandoned. In September 2017, it was re-introduced to the market by *CoinHive*, which was soon followed by competitors, such as *Crypto-Loot*. Both websites provide developers with APIs to implement browser mining into their websites. CoinHive works by embedding a Monero JavaScript miner into the webpage and using their visitors' CPU to mine the cryptocurrency, while the visitor is consuming the content of the webpage. Through this process the entity that embedded the JavaScript splits the mined coins with CoinHive, thus generating revenue.

Besides the JavaScript Miner, CoinHive launched two other separate services that help prevent Spam while mining Monero. Proof of Work Captcha, offers a captcha service where users need to calculate a certain number of hashes, before submitting a form. The other service, Proof of Work Shortlinks, provides an URL redirecting service, where users' CPUs have to also perform some mining operations before being redirected.

In-browser mining is considered to be an abuse unless users' give their consent. Mining scripts can be injected in a webpage via a multitude of attacks:

- The website administrator can inject the miner in the website without informing the users;
- JavaScript from third parties can be served within webpages. This could be via ads from an ad network, accessibility tools or tracking and analytics services;

- Browser extensions can include code to mine cryptocurrency unbeknownst to their users;
- Breached servers that are used by sites, extensions, or scripting services can be used inject mining code in such services;
- Man-in-the-middle attacks in open networks can easily inject mining code in every page users of that network access.

In terms of profitability, CoinHive developers estimate a monthly revenue of about 0.3 XMR, Monero's currency, which amounts to about 47.2 USD at the time of writing, for a website with 10-20 active miners [2]. In [8], the authors tested the profitability of the system during the experimental period of about 3 months. The website in question accumulated 105580 user sessions for an average of 24 seconds per session, with a revenue of 0.02417 XMR, which is worth about 3.84 USD. It was estimated that similar usage with traditional ad-based models would have been 2 to 3 times more profitable.

American news and opinion website Salon [9] was reported as the first website with a large audience to adopt this alternative as an option for users of ad-blocking software [10].

2.2 Cryptocurrencies

The idea of detaching cash from the organizations that control and police it has been around for a long time. Being highly dependent on these institutions generates centralized break points for the whole system, as it was shown by the financial crisis of 2007/08. When Satoshi Nakamoto first proposed *Bitcoin*, it was based on the premise that a purely peer-to-peer version of electronic cash could allow value transfers from one party to another eliminating the need for these centralized break points [1]. Other attempts at forms of digital currency had existed before, but none had the ability to be completely decentralized. This decentralization is particularly important since it enables cryptocurrencies to have a global reach, reduce transfer costs, remove points of failure and ensure transparency and trust.

Bitcoin uses cryptographic techniques and distributed CPU power to guarantee the decentralization and safety of the digital currency, hence the name cryptocurrency. With the expansion and progress of the project other cryptocurrencies emerged. Projects like *Litecoin* [11] were created via forking existing projects and subsequently altering certain aspects of them, whilst others were built completely from the ground up, e.g. *Ethereum* [12]. Litecoin in particular was formed via a fork of the *Bitcoin* Core client, and differs primarily by having a decreased block generation time, increased maximum number of coins and a different hashing algorithm.

Litecoin and *Bitcoin* both aim to replace traditional money, meaning their sole functionality is making payments without the need for a trusted third party to validate them. However, not all cryptocurrencies limit their features to value transfers. The *Ethereum* project for instance is also based on the blockchain, the same data structure as Bitcoin, but uses it in a completely different way. Ethereum uses the blockchain to provide a decentralized Turing-complete virtual machine [13][12]. The Ethereum Virtual Machine (EVM), which is able execute scripts using an international network of public nodes, allows entirely decentralized applications to be built on top of it. To achieve this, developers program their software using smart contracts. Smart contracts are high-level programming abstractions that are compiled down to EVM bytecode and deployed to the Ethereum blockchain for execution. Meaning Ethereum based applications can be used to transfer money, but also a lot more than that. One of the early applications that exemplifies this, is the popular browser game CryptoKitties [14], where players collect, trade and breed digital cats.

2.3 Blockchain

Nowadays *blockchain* has become an overused and abused buzzword in the world of finance. Few people understand how it works or what it represents and yet, mention the term regularly when talking about digital currency. To understand what blockchain is and why it came about, it is essential to comprehend why trust in financial transactions is a problem.

Consider two individuals A and B, that want to exchange value between themselves. When A sends X amount of value to B, there is a mandatory condition that needs to respected in order for the used system to work. This being that when value is transfered, both the sender and the receiver need to respectively lose and gain the accorded value, permanently.

Cash is the most ancient method of value representation, and still works to this day because it is relatively hard to replicate, meaning, that A cannot simultaneously give the same 100$ bill to two distinct individuals. However cash is not perfect, it deteriorates over time, requires physical presence to exchange and, although difficult, fake cash can be inserted into the system. Therefore, if A wants to send value to B but is not physically able to do so, it needs to rely upon financial institutions like banks to provide such service, thus inserting a third-party into the equation. This third-party is then responsible for verifying and validating the transaction, so that it respects the condition previously mentioned. As explained by Morgen E. Peck [15], when you use a check to purchase something, a series of agreements occur in the background between your financial institution and others, enabling money to go from your account to someone else's. Your bank can vouch that your money is good because it keeps records indicating where every

penny in your account came from, and when. This dependency on financial organizations to act as trusted neutral parties creates a breakpoint for the system, where human error has space to exist.

The *Bitcoin* paper [1] proposed a version of digital currency that could stand as an alternative way for people to store and trade value, without relying on the third-parties previously mentioned. The paper defines an electronic coin as a chain of digital signatures, where transfers are executed by digitally signing a hash of the previous transaction and the public key of the recipient, and adding these to the end of the transferred coin. The signatures could then be used to verify the chain of ownership. However this only provided part of the solution, since the recipient of said transfer could not verify that the same coin was not sent to other destinations, as part of a double-spending attack, and consequently a third-party would still be necessary to validate the transactions of such system.

To fix this problem, it was necessary to have an automatic way to guarantee that the previous owners of a coin did not sign any earlier transactions. Nakamoto understood that to achieve this behavior, it was necessary for the system to be aware of all transactions, so that only the earliest transaction could be considered valid. In a model where a trusted party exists, that entity is tasked with verifying all transactions and deciding on which arrived first. Without having a trusted party, it is necessary to publicly announce all transactions, and letting a system of participants agree on a single history of the order in which the transactions were received. It was this premise that led to the creation of the timestamp server that is now know as the blockchain.

In simple terms, blockchain is a list of records, called blocks, which are linked and secured using cryptography. It works by timestamping a hash of a block of items and widely publishing the hash to all the participant nodes. The timestamp proves that the data existed at the time it was published, and includes the previous timestamp in its hash, forming a chain, as seen in Figure 2.1. It starts with a genesis block, which is a block with no parent blocks, and grows from there creating a *Merkle* tree, where each following block has exactly one single parent block. The ledger is then formed by a continuously growing stack of immutable records. Updates are made by adding more information, instead of altering what was previously written.

It is thanks to these characteristics that *Bitcoin* and many other digital assets can have a distributed ledger, recording and totaling all economic transactions done using the asset. Without a system, such as blockchain, that can work as a traditional ledger, it would be impossible to create a truly decentralized digital currency.

The concept later rose in popularity amongst investors, mainly due to the fact that compa-

nies started to realize that blockchains could have many more purposes than just trading digital currency. In March 2014, the possibilities for how *Bitcoin* could be used were greatly expanded by the addition of the OP_RETURN field, in *Bitcoin* Core version 0.9.0, allowing users to insert 40 bytes of arbitrary data in each transaction. It became possible to store more information in the *Bitcoin* blockchain than just currency transfers, effectively growing the prospect of using the blockchain as a distributed database. In *An analysis of Bitcoin OP_RETURN metadata* [16], it was detected a growth tendency in the number of blockchain-based applications that embed their metadata in OP_RETURN transactions. The conducted study identified that although in the first year of existence of OP_RETURN transactions, only a few hundreds of these transactions were appended per week, their usage had steadily increased since March 2015. Overall it was estimated that OP_RETURN transactions constituted $\approx 1\%$ of the total transactions in the blockchain, and occupied $\approx 0.3\%$ of its space. In [17], the authors point out that fundamental differences exist between storing data in a centrally controlled database and what a blockchain database can accomplish. Whilst in blockchains, transactions are accepted via a consensus mechanism, traditional databases have a central party that controls which transactions go through. Consequently, if using blockchain a user cannot be certain if a transaction will be accepted, even after issuing it. Furthermore, a yet unaccepted transaction cannot be retracted by the user, and may be appended to the blockchain at any point in the future. Accounting these characteristics it is easy to see why blockchain systems can be particularly useful as databases for immutable information like identity control, supply management or ownership of assets. An example of how this immutability can be useful is illustrated by the paper [18], that proposes a decentralized review system called *ReviewChain*, where users can submit and retrieve untampered online reviews of products.

Types of Blockchains

To start with, blockchain began as a public or *permissionless* network, where anybody could create an address and begin interacting with it. A good example of a *permissionless* system is the Internet, where anyone can connect at anytime. In a similar way, in a *permissionless* blockchain, any person, or entity can interact with other parties by creating an address on the network. Each party can choose to run a node for the blockchain and participate by verifying transactions or create smart contracts. Both *Ethereum* and *Bitcoin* are considered *permissionless* networks and employ an economic model that encourages individuals to run network nodes in exchange for tokens. This process in which computing power is used to obtain cryptographic assets, like *ether* or *bitcoin*, is called mining, and guarantees that enough nodes run the network, making

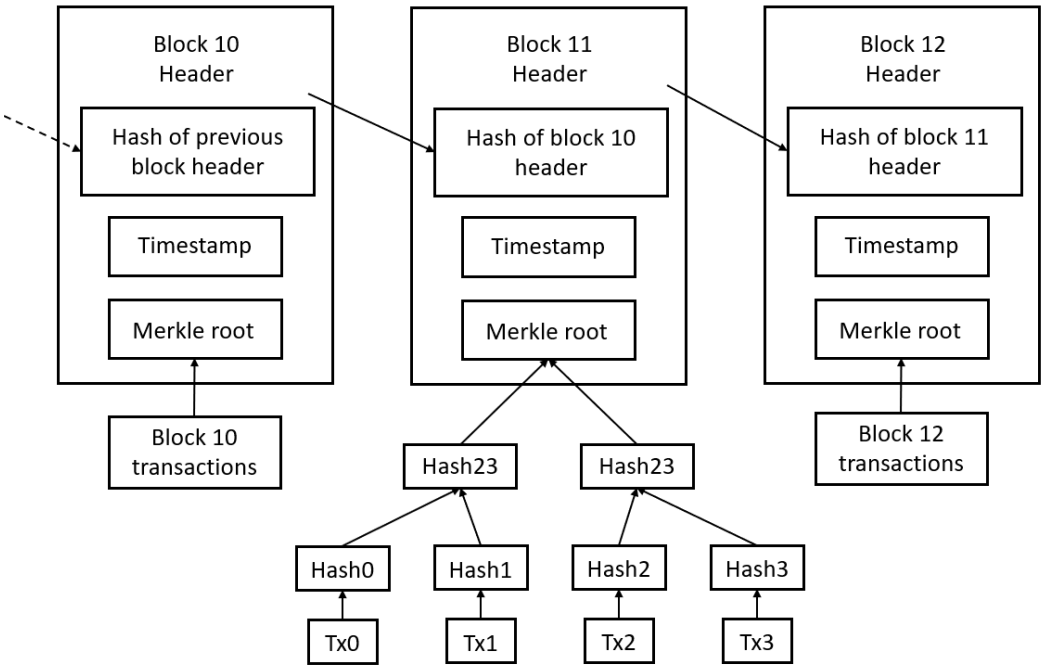

Figure 2.1: Overview of Bitcoin's timestamp server

it secure and usable. Public blockchains are gaining traction as a foundation for business-to-consumer and consumer-to-consumer use cases. According to [19], *permissionless* blockchains are commonly characterized by being decentralized and transparent systems, where users can stay largely anonymous. They also often suffer from performance and scalability issues, and encourage mining.

Although complete decentralization is what caused blockchain to be created, there are several use cases where blockchain networks can be useful and a minimal amount of trust between parties is present. For such scenarios a *permissioned* network is the best option. A *permissioned* blockchain is a closed ecosystem in which each participant is well defined. This type of blockchain is built to allow an organization or a consortium of organizations to exchange information and record transactions in a more transparent way. Consequentially, only pre-approved entities can run the nodes that validate transaction blocks and execute smart contracts on the blockchain. This makes it easy to share trusted information in a secure context, and with the confidentiality that businesses need to operate effectively.

Permissioned blockchains, can be separated into consortium and private blockchains. Private blockchains are differentiated mainly in terms of how *write* and *read* permissions. *Write* permissions are kept centralized to one organization instead of multiple, and *read* permissions may be public or restricted to an arbitrary extent. The differences between the two types are summarized in Table 2.1.

Table 2.1: Differences between blockchain types

Type	**Permissionless blockchain**	**Permissioned blockchain**
Consensus participants	All nodes	One/Multiple organization(s)
Read permission	Public	Public/Restricted
Write permission	Public	Restricted
Performance	Low	High
Decentralized	Yes	Partially
Security	Very hard to tamper	Hard to tamper
Example	Bitcoin[1]; Litecoin[11]; Ethereum[12]	Corda[20]; Hyperledger Fabric[21]

In recent years, consortium blockchains have been popularized through frameworks such as the Hyperledger Fabric [21] by Hyperledger [22], an umbrella project started by the Linux foundation, and Corda [20] by the R3 research consortium [23]. Multiple banks and large scale corporations have already started carrying out tests with the mentioned frameworks, as seen in [24][25][26][27].

2.4 Mining Pools

Mining for cryptocurrency is a high-risk high-reward activity. Mining a new block earns miners a large reward, but how often this happens is down to a matter of luck, and miners can only increase their probabilities by increasing their investment in more powerful hardware. Miners, seeking to reduce their variance and earn steadier incomes, take part in so-called pooling strategies where they jointly mine for a certain cryptocurrency. Participating in a pool is called pool mining and mining alone is called solo mining.

In these mining pools, whenever a participant mines a new block, the reward is shared with all the participants. Consequently, pools require a trusted operator to monitor participation and manage the allocation of rewards. Upon one of the pool participants mining a new block, the operator receives the block reward and then allocates the corresponding percentage among the participants based on how much work they contributed.

However, equating the earned percentage of the reward with the amount of work done by

the miner, is not a trivial task and requires constant monitoring of the effort of each participant by the pool. Unless miners are assumed to be honest, simply asking miners to report their effort leads to free riding. Riders will claim to have done work even if they have not. To overcome this problem, miners instead demonstrate their effort by submitting partial proofs-of-work called *shares*, which are simply block hashes that satisfy a lower difficulty parameter. Shares are a "near-solution" to the original computational puzzle, e.g. in a puzzle that requires the hash that solves it to start with 10 *0s*, a share would be requiring less *0s* at the start of the hash.

To prevent pool participants from stealing the block-mining reward whenever they find a full solution, the block owner identity is incorporated into the proof-of-work. Pools only accept proofs-of-work, partial or complete, that incorporate the identity of the pool as the block owner. Otherwise, miners could submit only partial proofs to the pool and send their complete proofs to the Bitcoin network for a solo reward.

2.5 Distributed Consensus

In a ledger registered in a blockchain, where the exact same single transaction history needs to coexist in all the nodes of the network, inconsistencies are to be expected. Reaching consensus among untrustworthy nodes is problem similar to the *Byzantine Generals* problem, first described in [28]. It consists in a group of generals, each commanding a portion of the Byzantine army, encircling a city. The generals have to formulate a plan for attacking the city, that in its simplest form, can consists of attacking or retreating. Each general can decide independently only whether to attack or retreat. The important thing is that every general agrees on a common decision, for a halfhearted attack by a few generals would lead to the worst possible scenario. It is through distributed consensus protocols that the participants of a blockchain network are able to maintain consistency.

In the case of *Bitcoin* and similar cryptocurrencies, each new block, with the exception of the genesis block, has a single parent, yet can temporarily have multiple children. This phenomenon, called forking, happens when not all the nodes in the network have the same history after a certain point, and is caused by either an attack or just naturally when different blocks are discovered, almost simultaneously, by different miners. Eventually, when the participants resolve the conflict, the alternative blocks either become part of the single chain of blocks or are rejected. Further in this section, several approaches to handle these conflicts are analyzed in detail. Table 2.2 shows a comparison between the referenced protocols and further extensive research about this topic can be found here [29].

Table 2.2: Consensus protocols comparison

Protocol	**PoW**	**PoS**	**DPoS**	**PBFT**
Energy Efficient	No	Varies	Varies	Yes
Tolerated faulty nodes	$< 50\%$ of total computing power	$< 50\%$ stake	$< 50\%$ stake	$< 33.3\%$
Example	Bitcoin[1];	Cardano [30]	Bitshares	Hyperledger Fabric[21]

2.5.1 Proof-of-Work

Proof-of-Work (PoW) is a consensus strategy used in the *Bitcoin* [1], *Litecoin* [11] and many other cryptocurrency networks. In a decentralized network, someone has to be selected to record the transactions. The easiest way would be random selection. However, random selection is vulnerable to attacks. So if a node wants to publish a block of transactions, a lot of work has to be done to prove that node's honesty. Work meaning computing power spent executing calculations.

In PoW, each node of the network has to calculate a hash value of the block header. The block header contains a nonce and network participants change the nonce frequently to get different hash values. The consensus requires the calculated value to be equal to or smaller than a certain given value. When one node reaches the target value, it broadcasts the block of transactions to other nodes and all other nodes must mutually confirm the correctness of the calculated hash value. If the block is validated, other nodes append this new block to their own blockchains, maintaining consensus across the network. This mechanism, creates a safe way to randomly select a leader to publish each new block. As a consequence, nodes have a chance of being selected proportional to the computational power they have.

In *Bitcoin* and other PoW based digital assets, users running the machines that act has nodes in the network, using their computational power to calculate hash values in exchange for tokens, are called miners.

When a conflict happens, as previously described, branches are generated. However, it is unlikely that two competing forks will generate next block simultaneously. PoW solves this conflict by judging the chain that grows faster as the authentic one. Consider two branches created by simultaneously validated blocks A and B. In this situation, miners keep mining their blocks until a longer branch is found. If A forms a longer chain, the miners on B switch to the

longer branch. Therefore, as long as at least 51% of the computing power of the network belongs to honest nodes, the honest chain will grow faster than the competing enemy one, making the network safe. Consequentially, PoW is very computationally intensive, brute-forcing the safety of the network through computational power, and wasting a lot of resources in the process.

2.5.2 Proof-of-Stake

Proof-of-Stake (PoS) is an alternative to PoW, designed with the belief that people with more wealth on the line would be less likely to attack the network. It came about due to the excessive resources required by the PoW protocol to elect a random node to publish the next block. In a PoS protocol, rather than miners investing computational resources in order to participate in the election process, they instead run a process that randomly selects one of them with a chance proportional to the stake that each possesses according to the current ledger. Essentially, the stakeholders themselves become responsible for maintaining the blockchain, and work, along with the corresponding rewards, is assigned to them based on the amount of stake they possess. This alternative selection mechanic should create no further "artificial" computational demands on the stakeholders.

At first sight this sounds ideal, but realizing such a Proof-of-Stake protocol involves a number of definitional, technical, and analytic challenges. The general concept is not secure by itself, and several PoS protocols have serious shortcomings. For instance, most variants are vulnerable to the *nothing at stake* and *long range* attacks, which considerably impact security in the blockchain [31].

In a nutshell, the *nothing at stake* vulnerability, is that if you have nothing at stake, then you lose nothing by behaving badly. In the event of a fork, whether the fork is accidental or a malicious attack, the optimal strategy for any miner is to mine on every chain, so that the miner gets their reward no matter which fork wins. For example, consider a situation where an attacker sends a transaction in exchange for some digital good, and that transaction gets registered in block x. The attacker can then start a fork of the blockchain from the block preceding the transaction, $x - 1$, and send the money to themselves instead, and even with 1% of the total stake the attacker's fork would win because the best option for all miners is still to mine on both.

The *long range* attack happens when an attacker is able to compromise the private keys of older accounts which held a large stake in previous blocks, but have no stake in the current block. The attacker can then use those accounts to alter the blockchain history, creating a fork from an already generated block. For instance, an account that had 30% stake at block height

h and no stake at block height $h+1$ can still use his 30% stake to create a fork height h.

Some PoS inspired protocols, that have been designed with the endeavor of addressing the these security challenges, are Ouroboros [32] used as the base for the cryptocurrency Cardano [30], and Casper [33]. Casper will be the future protocol for the Ethereum blockchain as it is in the process of moving away from Etash, a PoW protocol [34]. A variation of PoS called *Delegated Proof-of-Stake* (DPoS) also exists. The major difference is that PoS is direct democratic while DPoS is representative democratic. In DPoS stakeholders elect their delegates to generate and validate blocks. This translates into a need for significantly fewer nodes to validate a block. The block can be confirmed quickly, leading to the quick confirmation of transactions. Furthermore, the parameters of the network such as block size and block intervals could be tuned by delegates.

2.5.3 Byzantine Consensus

In computer science *Byzantine Fault Tolerance* (BFT) is the characteristic which defines a fault-tolerant distributed computing system, where components may become faulty, and there is no reliable information about which components are faulty. This directly correlates to what blockchain systems have face in order to achieve coherence between all nodes.

Several Byzantine consensus protocols have been developed, the most prominent one being the *Practical Byzantine Fault Tolerance* (PBFT) protocol [35]. PBFT is an algorithm designed to tolerate the aforementioned Byzantine faults. Much like the *Byzantine Generals* problem, the objective is having each node reaching its own individual conclusion and then having the whole system act based on the majority decision. Clients send requests to a replicated service to invoke operations and wait for a reply. The replicated service is implemented by *replicas*. The replicas move through a succession of consecutively numbered configurations called *views*. In a view one replica is the *primary* and the others are *backups*. The primary of a view is replica such that $p = mod|R|$, where v is the view number. If it appears the primary has failed view changes are carried out. Clients and replicas are non-faulty if they follow the algorithm which can be divided into five phases: request, pre-prepared, prepared, commit and reply. And proceeds as follows:

1. Client sends a request to the *primary*. The *primary* can then validate the message and propose a sequence number for it;

2. *Primary* sends pre-prepare message to all replicas. This allows the backups to validate the message and receive the sequence number;

3. All functioning replicas send prepare message to all other backups. This allows replicas to

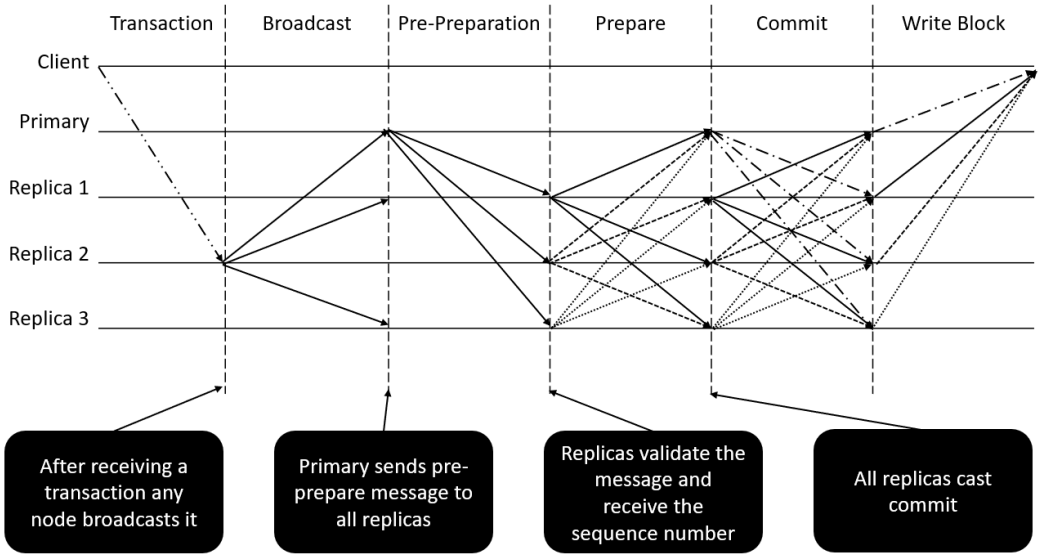

Figure 2.2: Phases of PBFT

agree on a total ordering;

4. All replicas cast a commit. The replicas have agreed on an ordering and have acknowledged the receipt of the request;

5. Each functioning replica sends a reply directly to the client. This bypasses the case where the *primary* fails between request and reply.

The algorithm ensures safety by guaranteeing that all non-faulty replicas agree on a total order for the execution of requests despite failures. In case the *primary* replica is faulty the client uses a timeout. When the timeout expires, the request is sent to all replicas. The described process is illustrated by Figure 2.2.

Given these assumptions, for the protocol to execute correctly, less than a third of all replicas of the service can be faulty at any moment. Thus, the total number of replicas N must be $N \geq 3F + 1$, where F is the maximum number of faulty replicas. PBFT requires a determined number of nodes to exist beforehand, making it best suited for permissioned systems.

Actual systems that implement PBFT or one of its variants are not very common. In fact, BFT-SMaRt [36] is one of the few projects that was developed before the interest in permissioned blockchains surged around 2015. There is widespread agreement today that BFT-SMaRt is the most advanced and most widely tested implementation of a BFT consensus protocol available.

2.6 The Scalability Trilemma

For any method of holding value, the ability to scale with demand is crucial. Bitcoin's premise to replace the existent fiat currencies with a digital and unregulated form of currency, was followed by a respectable solution that changed many of the existent paradigms associated with the concept of digital money. This solution, however, failed to meet part of the implications of said premise. Replacing the existent economic system demands that the new system has the ability to handle a similar or greater volume of transactions than the current one. This, along with the eventual demise of cash would require a cryptocurrency such as *Bitcoin* to be able to process a number of transactions per second, equivalent to the global number of physical and electronic transactions that happen in that time frame.

The present technology and algorithms used in the cryptocurrency field are far from achieving those numbers. For instance, as of April 2018, *Bitcoin* can only handle around 7 transactions per second (TPS). *Ethereum* and *Litecoin*, two of the following digital assets in terms of market cap, can handle around 15 and 50 TPS respectively, which is a considerable improvement in comparison to Bitcoin, but still far from achieving the same throughput as VISA with around 50,000 TPS. Achieving a similar or greater TPS than *VISA* would be trivial if these cryptocurrencies where also based on a centralized system, but being decentralized is essential for this type of asset. Hence, achieving scalability and decentralization without compromising on either, is a big challenge since they mostly are inversely proportional.

It is particularly hard to scale blockchains to the point where they can process more transactions than what a single computer can process in that network. To understand why this is the case, consider the function $O(n)$ which outputs the maximum number of transactions a node n can process. In an optimized blockchain system, the maximum number of transactions a network N can process will be equivalent to the average computational power of the nodes that constitute the net work, $O(n)$, and therefore $O(N) \leq O(n)$. This happens due to the synchronism that is required amongst all nodes to maintain consensus in the network.

Taking into account the earlier assumptions, there are a few basic scaling solutions that can be put in place, but have serious drawbacks associated. *Super-big blocks*, consists in the idea that by scaling the size of the transaction blocks, nodes can spend more power on handling transactions and less on calculating the next hash. E.g. *Bitcoin Cash* is cryptocurrency created through a hard fork of *Bitcoin* where the block size was increased from 1Mb to 8Mb. This works for small block increases, but super-big blocks would involve taking this into an extreme, meaning the amount of computing power necessary to process such blocks would be far greater than what a regular computer can accomplish, limiting the decentralization of the network.

Another alternative is having *multiple coins* for the system. Consider that a network N has a maximum TPS of $O(N)$. If the workload of N is split across 100 networks each with its individual coin the total TPS would increase also by a factor of 100 becoming $100 \times O(N)$. Although this can technically be considered a valid scaling solution, there are serious drawbacks to it. Besides the obvious resultant segmentation of the network, if N's workload is split across a 100 blockchains, then the number of miners would also be split across 100 blockchains. This would mean that, for the same total amount of miners, either there would be very few nodes running each blockchain, compromising its security, or alternatively *merge mining* could be allowed, so that each miner could mine every blockchain, but that would turn the total TPS of N back into $O(N)$.

These strategies effectively illustrate that a trilemma exists. Between the *security*, *scalability* and *decentralization* of a network, it is often possible to achieve two of them together at the expense of the third option. For example, the *Bitcoin* network is highly secure and decentralized, but suffers greatly from scalability issues. When higher scalability options are presented as possible solutions for Bitcoin, the network ends up suffering in terms of security, as analyzed here[37]. On the other hand, as previously explained, a network can easily be secure and scalable, e.g. *VISA*, at the expense of decentralization, or be scalable and decentralized but lose on security, e.g. *multiple coins*.

Other strategies to improve the scaling issue have also been designed. *Segregated Witness* or SegWit [38], is a soft fork change in the transaction format of *Bitcoin*, that helps mitigate the blockchain size limitation problem that reduces *Bitcoin* transaction speed. It splits the transaction into two segments, removing the unlocking signature (witness data) from the original portion, and appends it as a separate structure at the end. The original portion continues to hold the sender's and receiver's data, and the new structure contains scripts and signatures, working as the witness. This modification in addition to extending block size, also secures the network against malleability weaknesses. SegWit is not exclusive to Bitcoin and has been implemented in many other cryptocurrencies, including *Litecoin*.

Another possible future addition to Bitcoin is the *Lightning Network* [39]. It is designed to operate on top of a blockchain, enabling instant transactions of small amounts of currency between participating nodes. Normal use of the network consists of opening a payment channel with another node, by committing a funding transaction to the relevant blockchain. From the opened channel, the user can make instant transactions that are registered on the channel's funds, but are not broadcast to the blockchain. If person A opens a channel with B, and B opens a channel with C, as long as there are enough funds in the channels, A can also make

micro-payments to C without opening a channel directly to him. Once one of the participating nodes closes the channel, the result of all the transactions between the participants is sent to the blockchain as a single transaction. This can theoretically largely improve transaction volume, but there are some issues associated with the concept. For example, nodes on *Bitcoin*'s lightning network are required to be online at all times in order to send and receive payments, making them susceptible to hacks and thefts.

Sharding, is one of the options the *Ethereum* network development team is considering, as a means to improve scalability. The original concept comes from relational databases, as a form of partitioning the database to make access times shorter. In terms of the *Ethereum* blockchain, the goal is to split the network into various smaller networks, called shards. The shards all originate from the same root, and nodes within each shard can easily communicate with one another [40]. Splitting the network into shards, harnesses similar benefits to the *multiple coins* strategy, but also similar hindrances. One of such is inter shard communication, since each individual shard is an isolated network, nodes in separate shards would have to go through some sort of protocol to exchange information.

2.7 Summary

This chapter bestowed an overview of the background and works that are related with the subject of this dissertation. The project presented in this document, a CryptoNight-based web miner, stands as an alternative revenue source to the models presented in Section 2.1. As such, it leverages from work previously done in the cryptocurrencies field, with this chapter focusing prominently on what a cryptocurrency is, the distributed consensus algorithms that ensure their decentralization, and the obstacles that limit their scalability. Additionally, given the nature of how web-miners work, this chapter also introduces what mining pools are, why they exist, and the advantages inherent to their usage.

Having knowledge over the work presented in this chapter provides a better understanding of the sociopolitical factors that affect cryptocurrency valuation and the viability services such as the one presented in this document.

Chapter 3

Conceptual Design

Considering the web-mining services currently present in the market, such as the ones mentioned in 2.1, and their shortcomings, the chosen project that is presented here consists in an alternative more flexible web-miner.

The presented project is aimed at web site owners, who control their server, and would like to implement cryptocurrency mining in their websites, with end-to-end control over the system. Alternatively, the presented solution could also be used by third parties, in a web-mining as-a-service fashion, where such individuals handle the server side tasks for multiple adherent websites, such as configuring mining pools, in exchange for a fee.

In this chapter the conceptual design of the project is explained in detail. Section 3.1 introduces the entities that are involved in the system and their respective roles. Section 3.2 explains a series of concepts which are essential for a complete understanding of the system and the reasoning behind some of its design decisions. Section 3.3 illustrates the component structure of the solution and details their purpose within the system.

3.1 Participating Entities

The presented solution requires the participation of six types of entities: *server*, *administrator*, *mining pool*, *visitors*, *web browsers* and the *blockchain* of the used currency. Figure 3.1 shows theses entities and their interactions.

Visitors are the individuals who visit the websites adherent to the service, through web browsers. They share the unused computational power of their CPUs, by participating in the mining process and consequently support the owners of the websites they are visiting.

Web browsers are computer software applications for accessing information on the World Wide Web. When visitors access a website using the developed project, the browsers run the

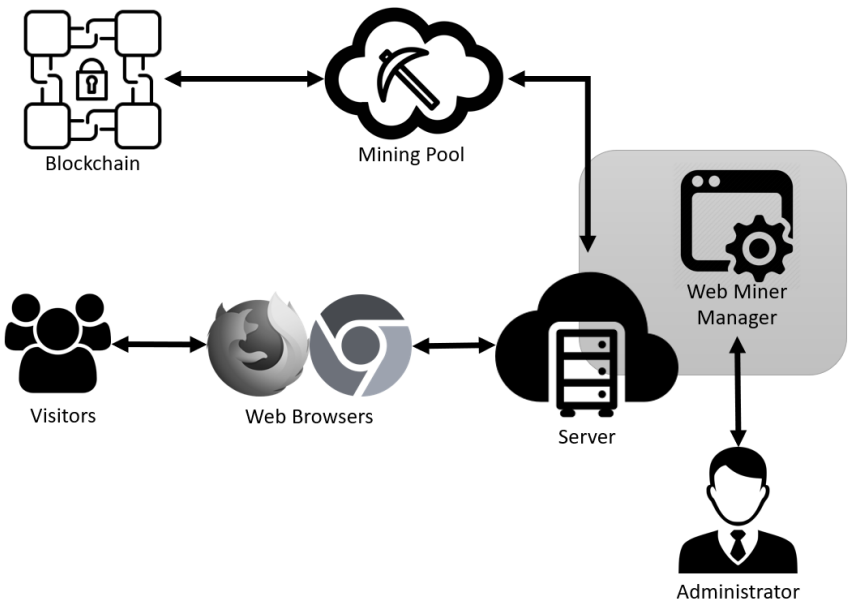

Figure 3.1: Entities that participate in the web mining process

JavaScript code that is responsible for the connection with the server and resolving the hashing problems required in the mining process.

Servers are the machines that run the application responsible for connecting the web browsers with the mining pool. This application is called *Web Miner Manager* and it communicates with one or more pools to get new hashing problems, called mining jobs, which are then distributed to the web browsers of the visitors.

Pools are third party services which run *mining pools* (see Section 2.4) where miners join their resources together and share their hashing power while splitting the reward equally, according to the amount of work they contributed to solve a block.

The Blockchain (see Section 2.3), is a distributed data structure, consistent of a growing chain of cryptographically connected blocks, which work as a ledger for all the transfers made using a determined cryptocurrency. Mining pools interact with a blockchain to submit a new block when one is mined by the pool, or get updates on the latest transfers and blocks submitted by other miners.

Administrators are the individuals with access to the servers and responsible for configuring and maintaining the *Web Miner Manager*.

3.2 Used Technologies

The developed project is based on a series of protocols and algorithms that are already present in similar systems. In this section, as a way of providing a better understanding of the choices

made for the design of the system, some background on those algorithms is presented.

3.2.1 Stratum Protocol

Stratum [41] is a protocol used to create lightweight clients for Proof-of-Work based cryptocurrencies. With Stratum, clients can work while just keeping the private keys, instead of having to locally keep the whole blockchain. Absence of the blockchain on the client side, facilitates a positive user experience, given that the client generates a very small footprint and has less demanding hardware requirements, while still providing high levels of security and privacy.

More technically, Stratum works as an overlay network on the top of the default P2P protocol of a cryptocurrency, providing a simplified API for accessing the blockchain stored on Stratum-based servers, thus hiding unnecessary complexity of the protocol.

The developed project is based on the Stratum protocol, which is an evolution of the *getwork* RPC method [42] used by a miner to get hashing work to perform. It came about due to the necessity of better supporting polled mining. With *getwork*, the block header was passed from the server to the client, without any transactions, and the only way to modify the block was through the nonce value. Consequently, the maximum that a client could do was to try all the nonce values before requesting more work from the server.

Stratum is a line-based protocol which means, it has elements that are delimited by the character sequence **\r\n**. It uses plain TCP sockets, with the payload encoded as JSON-RPC messages, which can work over HTTP and HTTPS. The client simply opens a TCP socket and writes requests to the server in the form of JSON messages finished by the newline character **\n**. Every line received by the client is again a valid JSON-RPC fragment containing the response.

This solution has considerable advantages over the *getwork* method:

- It is very easy to implement and to debug, because both sides are talking in human-readable format;
- Unlike many other solutions, the protocol is easily extensible to various cryptocurrencies without messing up the backwards compatibility;
- It uses JSON to format its messages, which is widely supported and current miners already have JSON libraries included;
- It enables *Long Polling* (see subsection 3.2.1).

One of the characteristics that makes Stratum very web-miner friendly is it running over HTTP. However since HTTP was designed for web site browsing where clients ask servers for

content, there are some intrinsic inefficiencies when using it for web mining where clients not only request information, but also need to send it.

These inefficiencies present when mining over HTTP, are solved by Stratum. For instance, if a miner wants a new mining job, it has to send a request through HTTP, but having the miners request this information when the mining pool knows more efficiently what each miner should be doing, is very wasteful. Particularly in the event of a new block appearing on the network and the miner not knowing about it, since it is connected to the pool instead of directly to the blockchain. Stratum fixes this problem by having the mining pool send information as it becomes available, and not miners asking for information periodically, regardless of its availability.

In [43], Recabarren, a penetration tester, analyzes Stratum Protocol vulnerabilities and suggests some solutions.

Long Polling

Stratum fixes the some of the limitations created by the request-response nature of HTTP, mentioned earlier, through *long-polling*. Long-polling is done by setting a high timeout limit on the server, purposely delaying the server from sending information until it has something to send. If such a timeout is not implemented, accessing even a simple URL page could take minutes to load because the server would be waiting for something to happen on its network.

When pools where first introduced in cryptocurrency mining, it became evident that it was necessary to choose between short polling intervals, which meant higher network loads and lower stale ratio, and longer intervals, which did not overload network and servers, but lead to a much higher ratio of rejected shares. This was resolved by using long-polling.

Without long-polling there is a lot of inefficiency in web-mining, because miners are requesting data that simply is not there and servers have to maintain those connections for as long as they are mining.

3.2.2 Hashrates

Cryptocurrency mining works as a competition, where miners compete to be the first to find a particular solution for a hashing problem. The rate at which a device solves these problems is called *hashrate*. The miner with the highest hashrate has the highest chance to win the competition, and usually those miners are run with GPUs or ASICs.

An ASIC (Application-Specific Integrated Circuit) is a chip specially designed for a specific application, such as resolving a particular hashing function. Therefore it is natural that it manages to achieve much higher hashrates than generic computer hardware, as seen for the case

of *Bitcoin* in Figure 3.1.

Table 3.1: Mining speed comparison for Bitcoin [44]

Device	**Mining Speed (KH/s)**	**Power Consumption (Watts)**
AntMiner S5	1155000000	590
HashCoins Zeus v3	4500000000	3000
AMD 5970	800000	350
Nvidia GTX-480	140000	250
Intel i7-3930K	98	200
AMD FX-8350	65	125

Considering that most popular cryptocurrencies use hashing functions which either have a vastly higher hash throughput on GPUs than CPUs, or already had ASIC miners developed for them, it is impractical to consider mining such currencies using a CPU-bound strategy like web-mining. Hence, web-mining services must focus on mining coins based on hashing algorithms that do not favor GPUs over CPUs, by a very large margin. For instance, Table 3.2, shows the hashrates of some CPUs and GPUs for Monero, a cryptocurrency frequently used by web-mining services.

Table 3.2: Mining speed comparison for Monero [45]

Device	**Mining Speed (H/s)**	**Power Consumption (Watts)**
AMD 5970	348	350
Nvidia GTX-480	337	250
SAPPHIRE RX VEGA 64	2020	130
Intel i7-3930K	338	150
AMD FX-8350	250	125
AMD RYZEN 1950X	1450	185

3.2.3 CryptoNote

CryptoNote [46][47] is an application layer protocol that allows for increased privacy in cryptocurrency transactions. It powers several decentralized privacy-oriented cryptocurrencies, *Monero* and *Aeon* being some of them, and achieves consensus using a *Proof-of-Work* 2.5.1 strategy, that has similar hashrates in GPUs and CPUs.

The original protocol, published in 2013, proposed a memory-intensive hashing algorithm, *CryptoNight*, which was designed to close the gap between CPU mining and that of GPUs and ASICs. However, since its release in 2014, ASICs have since been designed for the original CryptoNight algorithm. As such, the original hashing algorithm was a modified through a hard fork, making it incompatible with those ASICs.

CryptoNight

CryptoNight is a PoW algorithm that relies on random access to the slow memory and emphasizes latency dependence, where each new block depends on all the previous blocks. It was designed to make CPU and GPU mining roughly equally efficient and restrict ASIC mining, requiring around 2 megabits per instance.

It is able enforce these limitations, since:

- It fits in the L3 cache (per core) of modern processors;
- A single megabyte of internal memory is almost unacceptable for the modern ASICs;
- GPUs may run hundreds of concurrent instances, but they are limited in other ways. GDDR5 memory is slower than the L3 cache used by the CPU and remarkable for its bandwidth, not random access speed.

CryptoNight Lite is lightweight variation of CryptoNight, developed by the Aeon team with the goal of having a more mobile-friendly currency.

Given the limitations of in-browser mining, this project was developed to mine cryptocurrencies based *CryptoNight* and *CryptoNight Lite* PoW algorithms.

3.3 Component Overview

The component structure of the developed project can be seen in Figure 3.2. The system, is divided into three main parts, the mining pools, the servers hosting the *Web Miner Manager* and the visitors' web browsers running the miner JavaScript and user control interface.

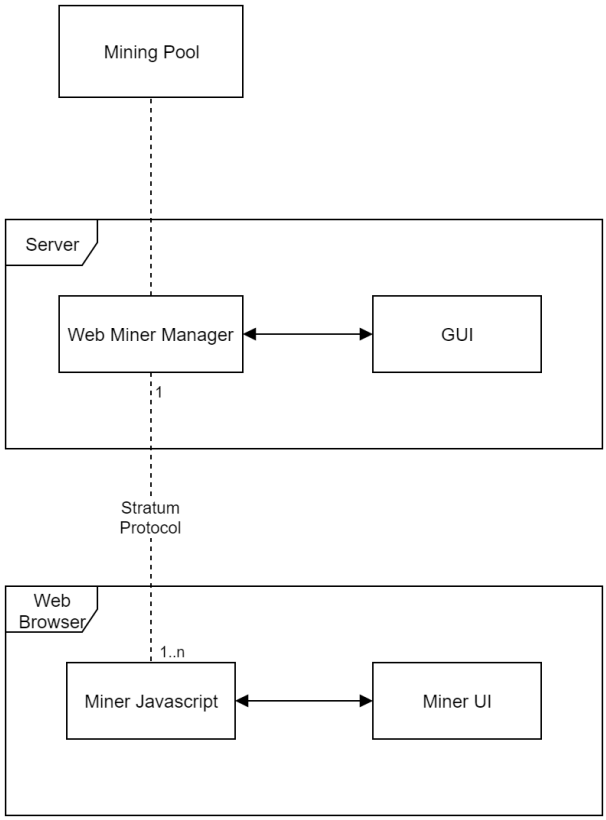

Figure 3.2: Overview of the component structure

In this system, all the interaction with the blockchain of the cryptocurrency being mined, is done by the mining pool in use. The *Mining Pool* works as a mediator, reading the updated status of the blockchain to get more work, which is then distributed by all the web-miners connected to the service under the form of, easier to solve, sub-problems. When a solution to such problems is found by one of the miners, it is sent to the pool, which is then in charge of verifying if that solution also is the hash required to mine a new block. If that scenario is realized, the mining pool publishes the new block to the blockchain, earning a reward that is subsequently shared among all the miners working on that mining pool.

The *Web Miner Manager* stands as the connection between the web-miners and the chosen mining pools. It is through the WMM that the server becomes available for connection by the web browsers. When a visitor accesses a compatible webpage, the JavaScript code in that web-page will automatically try to establish a connection with one of the server addresses designated in the code. If the WMM in that server was started and correctly configured, a connection will be established allowing the web-miner to start contributing.

Through a GUI (Graphical User Interface), show in Figure 3.3, besides starting and stopping the service, an administrator with access to the server can add/remove pools from the list of selectable pools that the program can use, select the pool to be used by default, configure

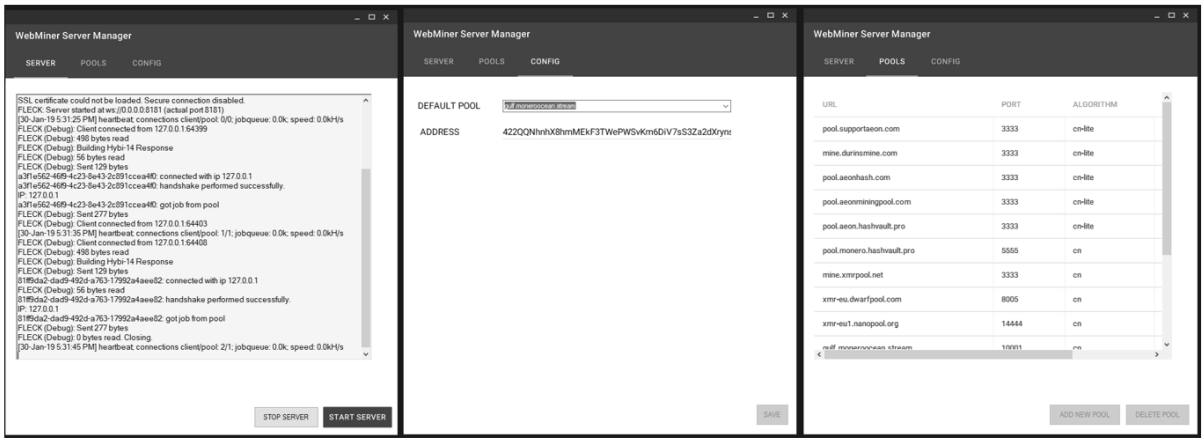

Figure 3.3: Graphical User Interface of the Web Miner Manager

the Monero or Aeon address to where the earn rewards will be sent, and monitor the current connections to web-miners. Being able monitorize the number of established connections is relevant since it gives the administrator an idea of how many visitors are adhering to the service and how the number of connections my be impacting the performance of the server.

In the event of a visitor accessing a webpage adherent to the service, the JavaScript code in that page initiates the Stratum Protocol by establishing a connection with the WMM. Once the handshake is formed, the Manager connects to the defined mining pool, to get work which is then redirected to the visitor. It is at this point that the visitor starts mining for the service.

The visitor can then see how many hashes his CPU has calculated and control, through the available UI in the website, the maximum amount of CPU usage he wishes to allow, while participating in the mining process or fully suspend his participation in it. The visitor can also access more detailed information, through a log of his connection status to the mining pool. Besides providing the aforementioned tools, this *Miner UI* is mainly responsible for keeping the visitor aware of what his computer is doing, and the purpose for doing so. The owners of the website can customize how they convey this message to the visitors, by editing the text in a pop-up windows that is shown by default when the mining process starts.

This architecture can theoretically support any cryptocurrency that can be mined using the CPU. Due to the large difference in performance of CPU mining in comparison to GPU mining for most cryptocurrencies, this project was developed with a focus on cryptocurrencies that try to mitigate that difference. The developed prototype was developed and tested to work with the cryptocurrencies Monero and Aeon, in particular. These cryptocurrencies are based on the CryptoNight and CryptoNight-lite algorithms, respectively.

3.3.1 Contributions

This design satisfies the requirements to achieve the objectives of the thesis, by:

- Mining Monero and Aeon, which are two of the most popular cryptocurrencies that don't favor GPUs over CPUs and have large user bases;
- Granting website owners more flexibility than the competition.

  The project follows a similar architectural pattern to other web-mining solutions, with the main difference being that it connects to external third-party mining pools instead of running a private one. This feature allows the administrator to have maximum flexibility. Meaning if he so chooses, a private pool could be implemented and used alongside third-party ones. If the system had been designed differently, and running a private pool was a necessity, it would stand as a hindrance rather than an advantage for websites with a smaller user bases, since running an end-to-end web mining service would render their pool hashing power exclusively dependent on their website's visitors alone, i.e., it would significantly reduce the pool's chances of obtaining any rewards.
- Costing website owners less fess than the competition.

  Since, a typical *Monero* mining pool like *NanoPool* takes around 1% of the rewards as fees, whereas CoinHive takes 30%.
- Enabling visitors to decide whether or not to use the technology, and giving them control over their CPU power usage.

  This is done with the addition of tools that ensure visitors' can have control over their participation in the system. It also detaches this design from its competition, as this can play a major role in gaining trust from the website's user base, and consequently more adherence.

3.4 Summary

To summarize, this chapter presents an overview of the conceptual design and component structure of the developed project. It consists of a web-miner, that under normal operation has six different entities partaking in the mining process.

The proposed solution is based on the stratum protocol, which improves upon the *getwork* RPC method. This protocol is line-based, uses play TCP sockets and runs over HTTP, making it web-miner friendly. Although HTTP usually works in only one direction, where the server

answers to content requests from clients, this protocol uses the long-polling pattern to establish a bi-directional communication between the server and the client, without increasing server load times. The ability to sustain open connections without extending server load times is essential for web-miners, as information needs to be sent back and forth during the process.

The web-miner was developed to work with CryptoNight and CryptoNight-lite cryptocurrencies. This choice was made due to the more favorable hashrates of the algorithm using CPUs, in comparison to other alternatives.

Chapter 4

Implementation

The developed project is designed to be an alternative revenue model for entities that already have running webpages. It aims to be easy to integrate with the existing code base and to require minimal maintenance after the initial setup. This chapter focuses on the implementation aspects of the developed system. It explains how the functionality is achieved, the internal structure of the code, and how some of the issues found during development were solved.

The remainder of this chapter is divided in two sections. Section 4.1 explains the internal construction of the server side of the system. Section 4.2 details the main components of client side of the system, which runs in the web-browsers of visitors.

4.1 Web Miner Manager Implementation

The *Web Miner Manager* (WMM) is the core component of the web-mining system, working as the middle man between the mining pools and client web browsers which do the actual mining work.

This program runs on a server, waiting for other peers to connect to it. It does so by opening a TCP server socket that binds to a particular port on the machine and listens for incoming connection attempts. When a new connection attempt is detected, it accepts the connection and creates a socket between the client and the server over which the client and the server communicate. Through this connection the WMM sends new mining jobs to the client web browsers, and the browsers reply with the results if a suitable solution is found. A mining job is a JSON message that includes the data that needs to be hashed and the target for the solution to be acceptable, for instance, the hash could need to start with five 0 to be valid.

On the other hand, when a new client is detected if a connection with a mining pool has not been established yet, the WMM connects to the default pool, selected from a list of pools

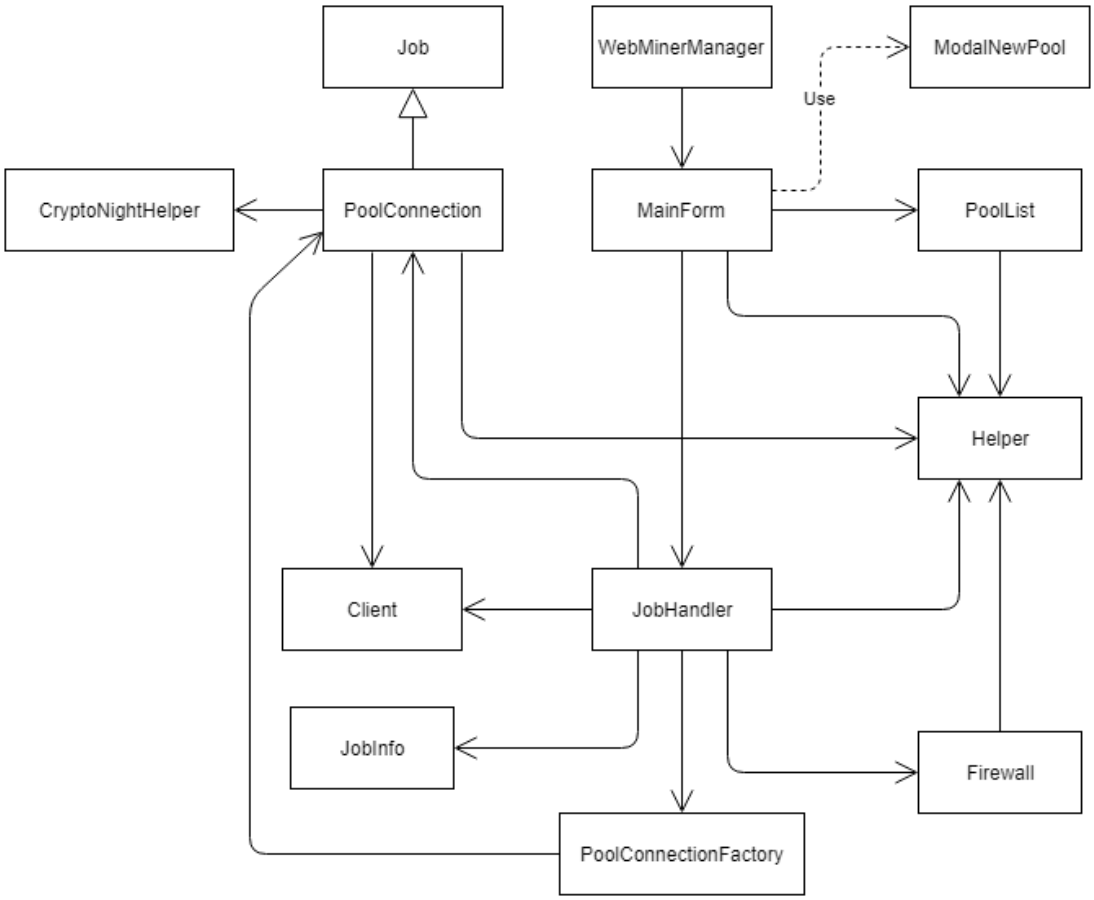

Figure 4.1: Web Miner Manager's simplified class diagram

present in the configuration options, using its specified credentials.

Because multiple clients are connected to the same pool, the WMM bundles these connections in batches of a preset size. This implementation in particular bundles clients in batches of 100, i.e. if there are 1000 connected clients, the WMM will have 10 pool connections. This is done because having too many connections or batches that are too large in size can result in connection difficulties and hashrate fluctuations.

This component was written in C#, using the *.Net Core* framework and *Windows Forms*. It is built on top of CryptoNoter's [48] implementation of the *Stratum Protocol* 3.2.1, but differs in how the mining pool credentials are stored and accessed during the connection process. On top of the implementation of Stratum, it has a Graphical User Interface (GUI) which is the access point for administrators to configure the service.

The remainder of this section covers in greater detail the central pieces of the WMM architecture, seen in Figure 4.1, describing their functionality within the system. Subsection 4.1.1 presents a detailed explanation of the *JobHandler* class and the server loop within it, which is responsible for handling all communications between the mining pool and the web-miners.

Subsection 4.1 explains what stratum extensions are and how they are used in this project. Subsection 4.1.3 introduces the developed GUI and its relevance for the project's functionality.

4.1.1 Server Loop

The server loop is a segment of code where the server enters a loop, waiting for requests from clients connected to it. The class *JobHandler* holds the function *RunServer()* which is the core function of the server, containing all the initial setup of the service and the the server loop. It is triggered when the *Start Server* button is pressed on the GUI, and runs indefinitely, until the *Stop Server* button is pressed.

```
Data: void
Result: void
load pools from file;
register pool connection callback functions;
create new WebSocketServer;
WebSocketServer initialization;
while server is not stopped do
    heartbeats++;
    start asynchronous task that completes after a set interval;
    calculate total of hashes;
    get all active pool connections pcc
    foreach PoolConnection pc in pcc do
        CheckPoolConnection(pc);
    end
    get all clients cc;
    foreach Client c in cc do
        if c.lifetime > GraceConnectionTime then
            if c.PoolConnection is invalid then
                DisconnectClient(c);
            end
        end
    end
end
WebSocketServer.Dispose();
```

Algorithm 1: RunServer() pseudo-code

Algorithm 1, describes in pseudo-code the steps taken during the execution of the server loop. When the administrator signals the WMM to start operating, it starts by loading all the available pools and reading the credentials of the pool that was selected by the admin. The callback functions are then registered to the object that holds the pool connection. Following that, a server socket is created and initialized, i.e. the actions *OnOpen*, *OnClose*, *OnError* and *OnMessage* have specific behavior assigned to them. It is the *OnMessage* action that holds all the communication between the server and the clients. Subsequently, the program enters an

asynchronous loop, where it waits for requests from new web-miners, establishes connections, and assigns work to them. During this loop the long pooling pattern, mentioned in Subsection 3.2.1 is applied. Upon identifying misbehaving clients, the connection to them is cut.

Besides running the server loop the *JobHandler* class holds other functions, essential for the control of the service, such as:

```
private void RemoveClient(Guid guid){...}
private void DisconnectClient(Client client, string reason){...}
private void PoolDisconnectCallback(Client client, string reason){...}
private void PoolErrorCallback(Client client, JsonData msg){...}
private void PoolReceiveCallback(Client client, JsonData msg,
    CcHashset<string> hashset){...}
```

The first two, as the names suggest, are responsible for removing and disconnecting clients connected to the server, respectively.

The last three are the callback functions assigned to *PoolConnection* objects created on the *PoolConnectionFactory*. They define the actions to be taken upon receiving certain messages from the pool. Function *PoolDisconnectCallback* is executed when a signal to *disconnect* is sent by the pool, disconnecting the corresponding web-miner. Function *PoolErrorCallback* is used by the pool to reply with the result of a submitted hash. If the hash is accepted, the message will be empty, if the hash is refused the message field will have details about the error in question. Lastly, function *PoolReceiveCallback* receives new mining jobs from the pool.

4.1.2 Stratum Protocol Extensions

This web-mining system was developed to be compatible with CryptoNight and CryptoNight-lite cryptocurrencies but, as explained in Subsection 3.2.3, although they are similar the algorithms still differ. Hence, in order to achieve the flexibility of working with both, using *Stratum protocol extensions* is necessary. These are a subset of additional parameters used for algorithm negotiation between the miner and the pool. The implemented set of extensions is based on XMRIG [49], and thus when adding a new pool to the system, the fields *algo* and *variant* are required by the form. Since these are just some extra fields, the extensions are backwards compatible with the regular Stratum protocol.

To each *job* object the pool/proxy also needs to add an additional *algo* field and an optional *variant* field. Typically, when connecting to a pool compatible with these extensions, the miner should send a list of supported algorithms. Having multiple algorithms in the list would mean

Table 4.1: XMRIG Stratum extensions [49]

Algo	**Variant**	**Result**
cn	-1	Auto-detect. Works only for Monero
cn	0	Original CryptoNight
cn	1	Monero7/CryptoNightV7
cn	xtl	Stellite (XTL)
cn	xao	Alloy (XAO)
cn	rto	Arto (RTO)
cn-lite	-1	Auto-detect. Works only for Aeon
cn-lite	0	Original CryptoNight-lite
cn-lite	1	Aeon7

that the miner can switch algorithms in run-time, which is not the case for our miner and therefore, when connecting to a pool the WMM only sends one value, either *cn* or *cn-lite*, in the *algo* field.

Table 4.1 shows how different combinations of these parameters correspond to mining different cryptocurrencies. The developed system only supports combinations with the algorithms *cn* and *cn-lite* and variants of -1, 0 and 1.

4.1.3 Web Miner Manager GUI

A Graphical User Interface (GUI) was developed for the system, using *Windows Forms*, with the aim of facilitating its accessibility. Through it, administrators can configure and monitor the WMM and its connections. The interface is divided in three main sections, as Figure 3.3 illustrates.

The *Server* panel gives access to the *Start Server* and *Stop Server* buttons, as well as a text box with logs of the status of the service and the peers connected to it. By using the text box, administrators can better diagnose possible server optimization issues through, for instance, having the understanding of how many miners are working for the service in a certain moment.

The *Pools* panel contains a list of all the saved pool credentials, and the options to remove or add new mining pool credentials. To add a new pool, the administrator is required to input the pool URL, the port, the algorithm, the variant, and if necessary, the password, as the credentials for that particular pool, which will then be saved to the list and available for selection under the *Config* panel.

The *Config* panel allows the admin to select the destination address for the mined currency, and select the default pool to be used during by the mining service. The WMM only accepts

Monero or *Aeon* addresses as valid addresses, given that the system was not tested with other cryptocurrencies.

4.2 In-Browser Miner Implementation

The developed project is a system that, in essence, can be divided into two parts, the server side, which consists in the previously detailed *Web Miner Manager*, and a client side, which is going to be presented in this section. The client side of the web-mining system consists of all code that runs in the visitors' web browsers when they access a webpage adherent to the service. It was developed using JavaScript, CSS and HTML.

The remainder of this section explains in greater detail the components of the client side of this system. Subsection 2 analyzes the scripts that run the web-miners, and the nuances of their implementation. Subsection 4.2.2 describes how the UI that controls the web-miner was implemented and the importance of its existence.

4.2.1 Miner JavaScript

The JavaScript code that executes the web-miner and calculates the results of the hashing problems required by the mining process, is contained in a file called *miner.js*. When a page with this code is loaded, the following JQuery call is made:

```
$(document).ready(function (){
    startMining();
});
```

This leads to the execution of the *startMining()* function, contained in *miner.js*. This function simply verifies if all the conditions necessary for the browser to mine are met, and if so, it starts the execution of the steps detailed in Algorithm 2.

The major requirement is that the browser must be compatible with *WebAssembly*. *WebAssembly* (Wasm) is a standard that defines a binary format and a corresponding assembly-like text format for executables that are used by web pages. Wasm is necessary to enable the JavaScript engine of a web browser to execute page scripts nearly as fast as native machine code.

Wasm compatibility is required, in particular, for the portion of *workerFunction()* that contains a script to calculate CryptoNight hashes generated using *Emscripten*. *Emscripten* is a source-to-source compiler that runs as a back-end to the LLVM compiler and produces a subset of JavaScript known as *asm.js*, but can also produce WebAssembly. This allows applications

and libraries originally designed to run as standard executables to be integrated into client-side web applications, like this one.

Although Wasm code runs in the same sandbox as regular script code it is not a full replacement for JavaScript. Rather, Wasm is only intended for performance-critical portions of page scripts, such as the mining operations ran by this service, where performance and efficiency are two major concerns.

Data: void
Result: void
if *Wasm not supported* **then**
  return void;
end
var logicalProcessors = numThreads;
while *logicalProcessors* > *0* **do**
  addWorker();
  logicalProcessors = logicalProcessors − 1;
end
establish connection with the WMM;

Algorithm 2: *startMining()* pseudo-code

Upon verification that the browser is Wasm compatible, *web workers* are created for each of the CPU threads available. When the browser is not compatible with Wasm, the service simply does not start mining, remaining inactive. A *web worker* is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page. The usage of web workers to perform the mining operations is paramount, as the user can continue to do whatever he wants in the page, while the web workers keep calculating hashes in the background. New web workers are created when the *addWorker()* function is called.

An auxiliary function called *workerFunction()* used inside *addWorker()*, came to be due to limitations imposed by Chrome and Edge web-browsers. These web browsers do not allow *web workers* to be loaded using scripts from a separate local file, meaning that in order to create a new workers with the WebAssembly necessary to calculate CryptoNight hashes, it was necessary to use a workaround to this limitation. The solution was to have an extra function hold all the code necessary for the *web worker* and create the new worker by converting it to plain text, as shown by the code below:

```
function addWorker() {
    var newWorker = new Worker(URL.createObjectURL(
        new Blob(["(" + workerFunction.toString() + ")()"], {
            type: 'text/javascript'
    })));
```

```
    workers.push(newWorker);
    newWorker.onmessage = on_workermsg;
    setTimeout(function () {
        informWorker(newWorker);
    }, 2000);
}
```

Once the web workers are created, the connection with the server is established. Once this connection is established the WMM, will provide mining jobs to each of the web workers through messages encoded in JSON. If one of the workers finds a solution it will use the same connection to communicate it back to the WMM.

4.2.2 Miner GUI

Another major part of the client-side implementation is the GUI, used to give the user information about the mining process, and control over his computer's participation in it.

When the webpage is first accessed, a modal window opens up by default, greeting the user. This window, illustrated by Figure 4.2, informs the user on how the web-miner works, what his PC is doing, its importance for the website owners, and how he can control how much he computational power he donates to the website. Optionally, the user can also access a log of the connections and hashes his computer is calculating.

Figure 4.2: Information window example

After the initial greeting the user can then use the website as usual or access the control panel if he deems it necessary. The control panel allows the user to quickly see how many hashes have been calculated, open the information window again, turn off the mining process entirely or throttle the usage of his CPU to his liking thorough a slider. Figure 4.3 presents an example of the side control panel.

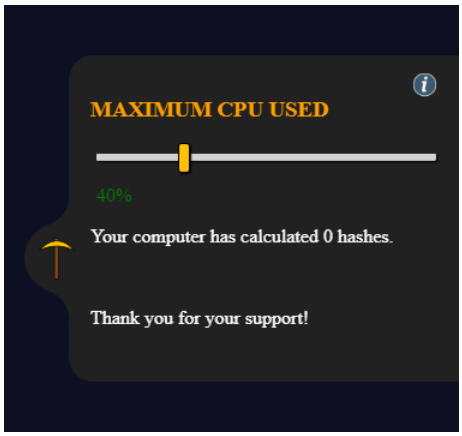

Figure 4.3: Side control panel

Since both these elements are built using only CSS and HTML code, they are easily configurable to fit a wide range of websites. The code that throttles CPU usage is part of the *miner.js* and therefore the UI can be modified independently without affecting functionality. CPU usage is limited by putting the worker thread to sleep before asking for more work, according to the code below:

```
function submit(){
    var t0 = performance.now();
    calcHash();
    var dt = performance.now() - t0;

    var sleept = Math.round(thrt / (100 - thrt) * dt);
    setTimeout(submit, sleept);
}
```

Where *thrt* corresponds to the percentage of available CPU power the user does not wish to be used, i.e. if the user limits CPU usage to 30% using the slider, $thrt = 100\% - 30\% = 70$, and as such the sleep time will equal $70/30 * dt$, with dt being the time that web worker took to calculate the previous hash. This approach is not perfect but is able to achieve a satisfactory emulation of CPU usage limitation, without requiring low level access to the hardware being used.

4.3 Summary

This chapter focused on the implementation of the solution presented in this document, explaining how the Web Miner Manager works, connects to mining pools and distributes mining jobs

to connected web-miners, and how the miner that runs on visitor's browsers and its respective UI were developed.

In Section 4.1, the internal component structure of the server-side code was analyzed, focusing mainly on the server loop. Stratum protocol extensions were also introduced, and their application to the solution explained. Finally, an overview of the GUI for the WMM was given, and its utility explained.

Section 4.2 explains the JavaScript code that runs the miner in detail. Some of the issues found during its development were identified and their solutions highlighted. Lastly the developed UI for the web-miner is shown and the CPU throttling algorithm clarified.

Chapter 5

Experimental Evaluation

This chapter presents the evaluation of the developed web-mining service, according to the objectives proposed in Subsection 1.3. Recall that the goal of the project is to provide an understanding of how cryptocurrency-based channels of revenue, particularly web-mining services, perform in terms of reliability and profitability, and how they can be improved. With this in consideration, this section aims to answer the following questions:

1. How well does the solution scale?
2. How accurately does the CPU limitation feature work?
3. What kind of operational costs are expected when running this solution?
4. How large does the user base need to be, for the service to break even?
5. How does the revenue generated by this solution compare with ad-based revenue models?
6. How does the solution compare to the competition?

Therefore, the evaluation provides an assessment of how the solution compares to its competition, presents a performance benchmark of the service, in terms of profitability and scalability, and infers how these results might translate in real-world scenarios.

The tests were performed using three machines, running a variable number of clients each, and *NanoPool* as the mining pool mining Monero. *Machine 1* runs an Intel i5-8250U CPU, *Machine 2* runs an AMD A10-7870K APU, and *Machine 3* runs an Intel i7-3770 CPU. *Machine 1* always doubles has the server running the Web Miner Manager. The results are inferred from extensive tests done with these machines, as their configurations represent those of an average consumer's PC.

Section 5.1 analyzes how the solution scales and spreads the workload across multiple clients. Section 5.2 examines the accuracy of the tool that gives the users control over their participation in the service. Finally, Section 5.3 evaluates the operational costs of the service and estimates how profitable it could be comparatively with advertisement-based revenue.

5.1 Scalability

For a web-mining service to be a reliable revenue option, scalability of the system to multiple simultaneous clients is crucial. This section provides an evaluation of the performance of the system with 1 client in comparison with 5 clients running on the same machine. Furthermore, the performance of the service with 15 clients running on 3 distinct machines is analyzed.

The goal of this section is to answer question 1, regarding the scalability of the system.

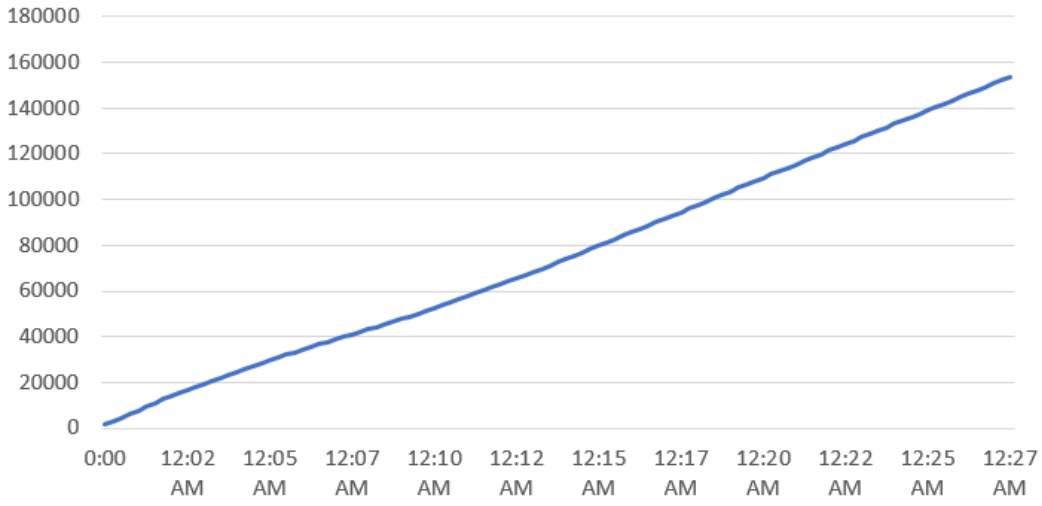

Figure 5.1: Hashes calculated over time by 1 client running on *Machine 1* at 100%

Figure 5.1 illustrates the results of our base test using 1 client running at 100% on *Machine 1*. The test was run for a duration of 28 minutes, during which the client calculated a total of 150714 hashes using *NanoPool* as the mining pool.

On the other hand, Figure 5.2 illustrates the results of running the same test but with 5 clients instead of 1. During this test total amount of hashes calculated by 5 clients was 172195. It is visible through the figure that each client calculated hashes at relatively the same speed as its peers, which allows us to infer that the distribution of the workload is spread evenly throughout all the connected clients.

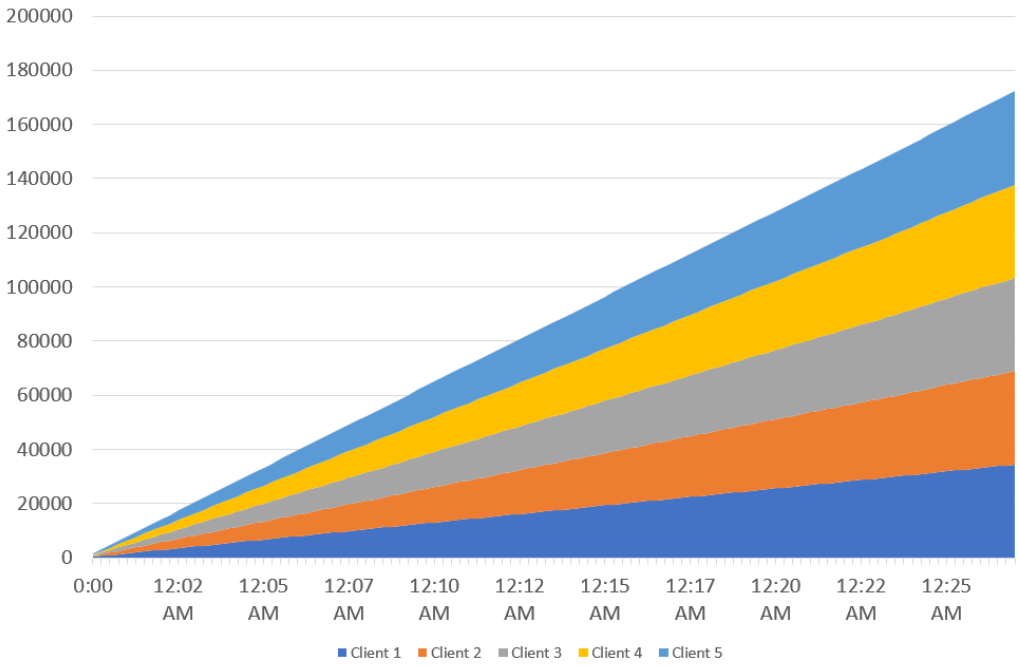

Figure 5.2: Hashes calculated over time by 5 clients running on *Machine 1* at 100%

When comparing the total amount of calculated hashes in 28 minutes between the tests, surprisingly, the test where the workload is spread across 5 clients had a better result, with a total of 172195 calculated hashes. However the difference of 21481 total hashes can be attributed to various variables, such as the fluctuations in the available CPU power of the used machine, variations of the difficulty from the mining pool, and connection speed.

Although the results of the tests that ran locally on a single machine were positive, testing the system running on multiple machines was indispensable. For that purpose, the service was tested running 15 clients running across *Machines 1, 2* and *3* at 100% CPU usage. The results of that test are shown in Figure 5.3.

The graph illustrates that each machine had its workload spread evenly by each of its 5 clients. Variations of the total amount of hashes calculated by each machine are visible, but easily attributable to difference of computational power between their CPUs. The results indicate that the service works and scales as expected with remote connections.

The results of these preliminary tests are positive and indicate that the system is able to scale while maintaining an even spread of the workload between all connected clients. Under normal conditions, the system should be able to handle websites with larger user bases and scale accordingly, up to a point in which the server might hit a bottleneck. At such a point, another server with a different IP address may be used in parallel with the original, by having the client-side code randomly choose which server to connect to.

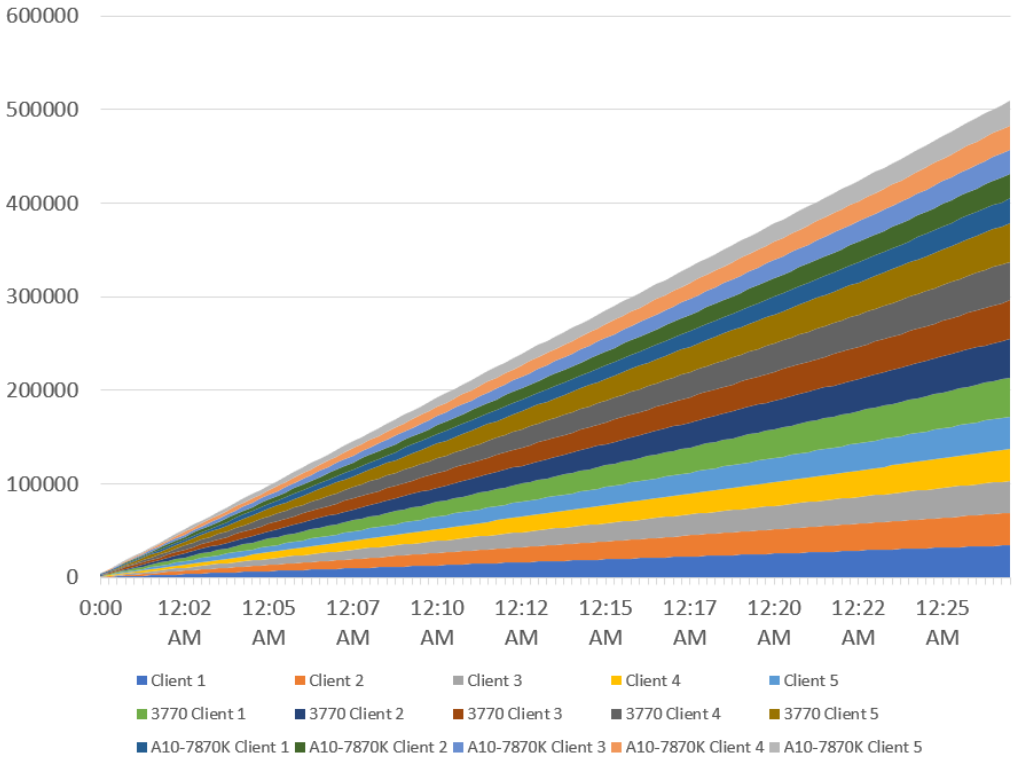

Figure 5.3: Hashes calculated by 15 clients running on 3 machines at 100%

5.2 User Control

In the presented prototype, users have control over their participation in the service through a slider bar. This slider allows users to choose the percentage of unused CPU power they wish to donate, or deactivate the service entirely, by putting the slider at 0%.

To validate that this tool was working appropriately, the test depicted in Figure 5.1 was repeated, but with the slider bar at 25%, 50% and 75%. The results of this test can be observed in Figure 5.4.

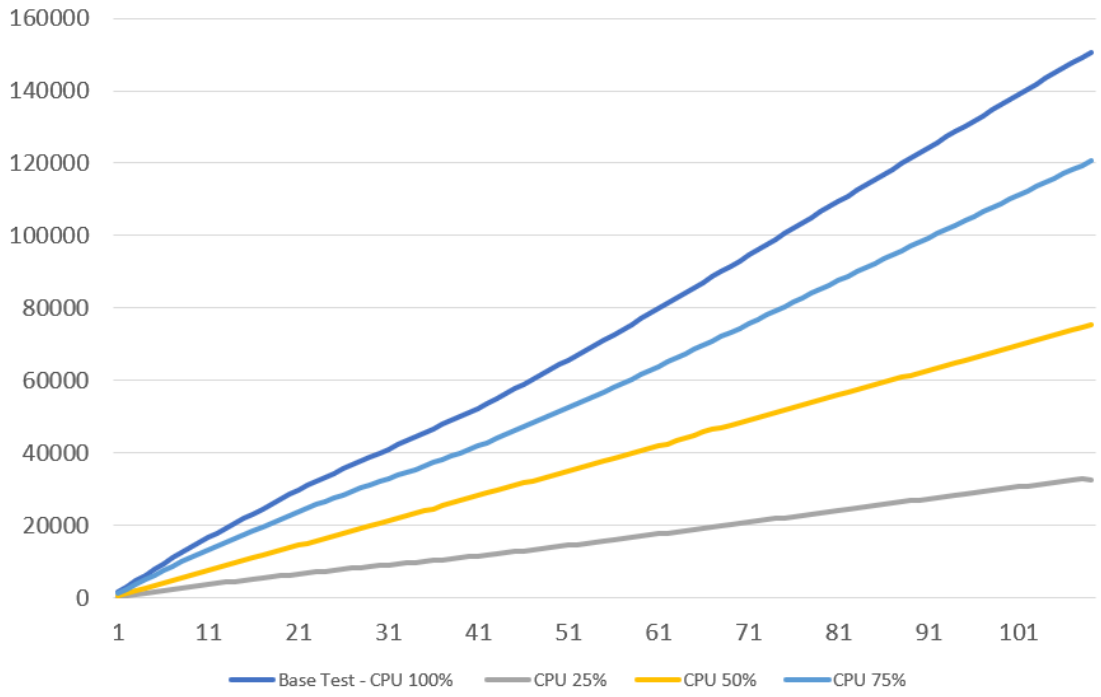

Figure 5.4: Hashes calculated over time by 1 client running on *Machine 1*

It can be deduced, through the analysis of the graph, that the tool is limiting CPU usage by the specified amount. The amount of hashes calculated per unit of time by the client running at 50% are roughly half of the ones calculated by the client running at 100% in our base test. In addition, the total amount of hashes calculated after 28 minutes by the client at 50% was 75209, compared with 150714 by the client at 100%, which further substantiates that the tool is limiting unused CPU power with precision. The tests that ran with the CPU at 25% and 75% showed that the hashes were being calculated at around 22% and 80% of the rate of the base test, respectively.

5.3 Revenue and Costs

The constant fluctuations of the value of cryptocurrencies make entirely relying on a cryptocurrency based revenue model, a daunting task for anyone expecting steadiness in their income, but when ads are out of commission, it might be the next best option. In this section we estimate how much revenue the developed solution could generate for websites with user bases of various sizes, and compare those results with the estimated revenue of an ad-based model.

In this estimation, the considered hardware operational costs are those of a basic AWS EC2 server, which are on average 40$ per month. The mining pool used in the previous tests, *NanoPool*, imposes a fee of 1% over the obtained rewards, which is considered in the estimation.

Furthermore the estimation of the rewards was calculated using *whattomine* [50], with a valuation of each *Monero* token at 52.49$, at the time of writing this document. Any costs associated with the conversion of cryptocurrencies to fiat currency are ignored.

Table 5.1: Estimated rewards generated by 1 user at 90h/s

Retention Time	**Pool Fee (XMR)**	**Rewards (XMR)**	**Rewards (USD)**
Hour	0.000000	0.000010	$0.00
Day	0.000002	0.000247	$0.01
Week	0.000017	0.001730	$0.09
Month	0.000075	0.007412	$0.39
Year	0.000911	0.090181	$4.73

The test illustrated by Figure 5.1 stands as a representation of the average user using the service, with an ideal contribution of 100% unused CPU power. The hashrate of the user represented by that test is around 90 hashes per second. Thus, for estimation purposes, it is considered that every user of the service would have an hashrate 90 h/s.

Another important concept to take into account is *retention time*. Retention time is the amount of time a user stays on the webpage, actively mining for the service. The estimated rewards generated by a single user in relation with to the retention time are shown by Table 5.1. This first example shows that any website that averages 1 connected user at the time, would have an estimated profit each month of −$39.61.

Expanding on the calculations of Table 5.1, Figure 5.5 illustrates how a growing user base would impact the profitability of the website. Through the analysis of the graph it is possible to estimate that a website breaks even using this service, when the average amount of users that are connected to it at any point in time is between 100 and 110.

5.3.1 Advertisements

Initial expectations indicate that ad-based revenue is more profitable than running a web-mining service on a website. But with the growing use of ad-blocking software, alternatives such as the one presented in this document are necessary.

Understanding the magnitude of the profit difference between using advertisements and a web-mining service, plays an important factor in the understanding of whether or not these

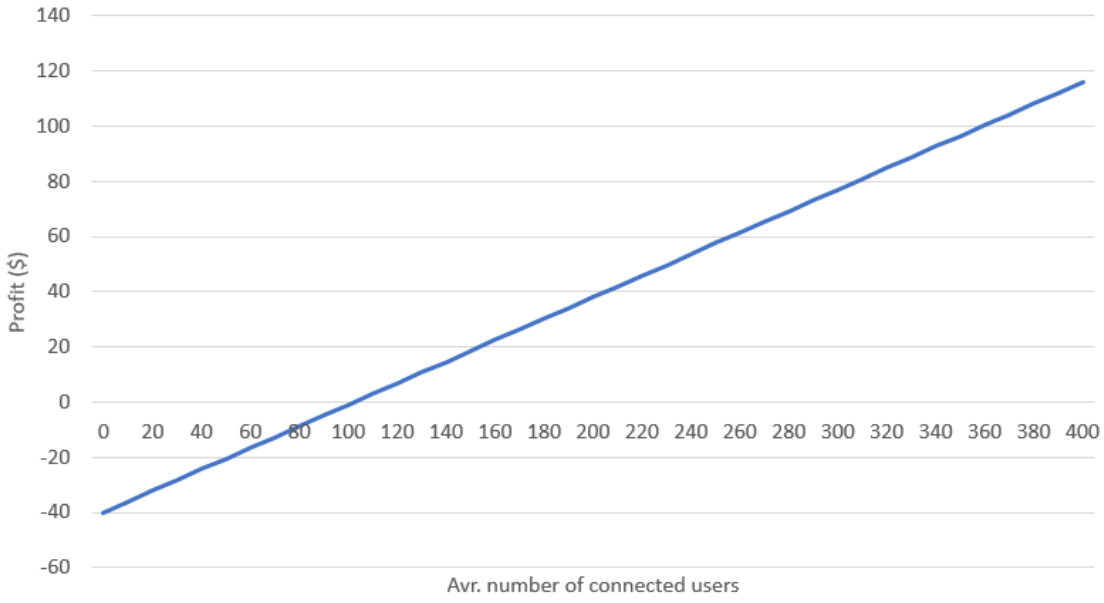

Figure 5.5: Service profitability in relation to average number of connected users

services will proliferate in the future. As such, considering the estimations shown in Figure 5.5, an assessment of how that hypothetical website would perform is made, using a *Cost-Per-Click* revenue model where website owners are paid every time a visitor click on an advertisement.

The amount of visitors V a website would have to get, to average a C amount of connect users during a time span T, is described by the following formula

$$V = TC/pt$$

where, p is the average number of pages a visitor accesses and t the average time a visitor spends on a page.

Assuming that for a particular website, t is 1 minute, p is 2, the percentage of visitors who click an ad is 2% and the payment when users click on an ad is \$0.30, it was calculated how the same website would perform using a *Cost-Per-Click* model in comparison with a web-miner model over a month, for various amounts of average connected users. The CPC Profit was calculated with the following formula

$$CPCProfit = V * clickrate * payment$$

Table 5.2: Profit comparison between CPC and Web-Miner revenue models

Avr. connected users per month	**Web-Miner Profit (USD)**	**CPC Profit (USD)**
10	−$36.10	$1296.00
40	−$24.40	$5184.00
70	−$12.70	$9072.00
100	−$1.00	$12960.00
130	$10.70	$16848.00
160	$22.40	$20736.00
190	$34.10	$24624.00

Some simple calculations based on the assumptions above, lead to the conclusion that a website that averages 1 user connected at all times during a month, has around 21600 visitors in a month and a profit of $129.6, which is still $129.21 higher that the web-mining service, even when ignoring the costs of running the server to host the *Web Miner Manager*. The results, shown in Table 5.2, make the disparity of profit evident, with the CPC model reaching over $12000 in profit before the web-miner model breaks even.

However, this is an hypothetical scenario, and either revenue model might have better or worse performance depending on the type of website, i.e. the disparity although undeniable, might be reduced in a website that has less visitors, but retains them for a longer time, such a browser game website.

5.3.2 Other Web Miners

Unfortunately *CoinHive* does not make available a profit estimation calculator for their service, but *CryptoLoot*, a competitor to *CoinHive*, does [51]. Using that calculator it is possible to estimate how the results shown in Figure 5.5 would compare if the website used *CryptoLoot* instead. The results of that estimation and how they compare to the developed solution can be seen in Table 5.6.

The estimation assumes that averaging 1 connected user per day corresponds to having 720 daily visitors that remain on average 2 minutes in the website, and that 1 month corresponds to 30 days.

Through the analysis of the graph, it is easy to understand that the main advantage of adher-

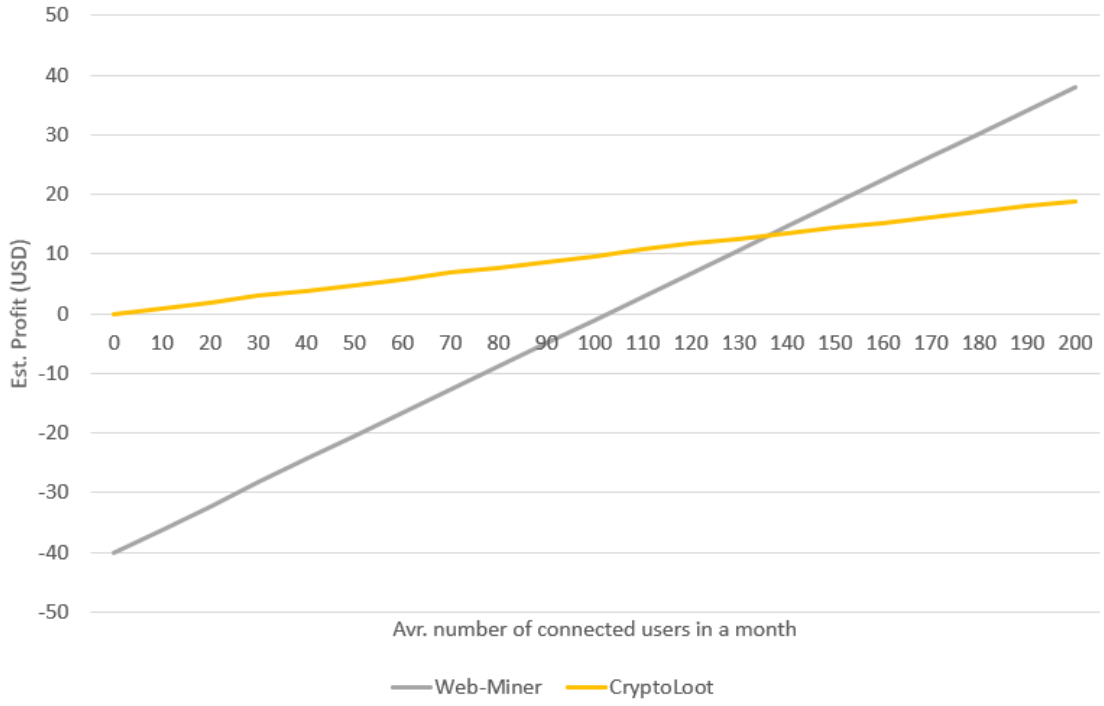

Figure 5.6: Estimated profitability of the solution vs CryptoLoot

ing to *CryptoLoot* is not having to handle the costs associated with running a dedicated server, whereas for websites with large users bases, running the service presented in this document becomes more profitable due to the smaller fee taken by the mining pool in comparison with the fee taken by *CryptoLoot*. Furthermore, the estimation also lets us assess that the presented service becomes more profitable than *CryptoLoot* for website that average between 130 to 140 users connected at all times.

5.4 Summary

To summarize, this chapter focused on the evaluation of the developed solution and analysis of how the obtained results correlate with the objectives proposed in Subsection 1.3.

Section 5.1 investigated how well the solution is able to scale, with tests examining how the Web Miner Manager performed when connected to clients on multiple machines. The positive results allowed to assess that the solution is able to scale enough to work with websites that have large user bases, with an appropriate spread of the workload between the connected clients.

Section 5.2 examined how precisely the tools given to the users, to control their participation in the mining process, worked. Once again the results we positive, with the tools performing as expected.

Section 5.3, assessed how the solution would perform in terms of generating revenue. Esti-

mations of the costs associated with running the service, how much profit it is able to generate and how that profit compares to the competition and a CPC revenue model, are made. The study concluded that to this day advertisement revenue models are still largely more profitable than web-mining services. This service in particular might be most useful for websites that have high retention times of their users and large user bases, but do not want to rely entirely on ads. For websites with smaller user bases, given the low profitability of the service it might be hard to reach the break even point, and thus using third party services like *CoinHive* or *CryptoLoot* might be a more adequate solution, since they have impose the extra costs of running a server.

Chapter 6

Conclusions

As ad-blocking software becomes more common, with works such as the Brave browser, embracing it as one of its selling points, companies that rely on advertisement revenue have started to look for solutions to mitigate their losses. With the most recent cryptocurrency boom, web-miner services resurfaced publicizing themselves as solutions, but a plethora of explorations by hackers have given this kind of services a bad reputation.

This project investigated how web-miners could be improved in terms of profitability and adoption by website visitors. As part of the investigation, a web-miner with a focus on transparency, user control and flexibility was developed. The profitability tests showed that for the current valuation of the cryptocurrency market, the profitability of web-miner services, including our prototype, is minimal.

Our estimations assessed that the developed prototype, improves upon other options, in terms of profitability, when the website using it has a very large user base. The prototype also included tools that give website visitors information and control over the mining process. The performed tests over these tools also showed positive results, that assured users' would be able to precisely control their participation. Furthermore the prototype ensured easy configuration and minimal maintenance through a simple GUI on the server-side application, which can be run indefinitely after the initial setup.

The developed work concluded that, as of today, web-miners are not valid replacements for traditional revenue models, but can be used as a side source of revenue when other options are not available.

6.1 Achievements

The evaluation of the performance achieved by our prototype showed that, it can achieve higher profitability than other alternatives, when either server hosting costs are negligible or the user base of the website is very large. The developed solution maximizes its accessibility, by guaranteeing that all interactions with it are done through graphical interfaces.

Finally, a big focus of the work was ensuring that visitors' awareness of the mining process and how they can control it, which we believe can play a major role in the adoption of these services.

6.2 Future Work

Although the presented solution is a complete web-miner system, there is room for many improvements. As future work, the prototype could be continued by addressing the following points:

- The hashes submitted by clients could be periodically checked by the server, as means to do some quality control of the connected clients and disconnect misbehaving ones;
- An identification system for clients could be created, so that users could be rewards by businesses for their participation. For instance, a stream service like *Twitch* could rewards users with their platform's currency;
- The Web Miner Manage could generated logs with general and per user statistics;
- As mentioned in Section 4.1.2, some mining pools support run-time algorithm switching. This feature could also be implemented in this project;
- Support for multiple owners could also be implemented, as a way of having one server running a single instance of the Web Miner Manager service multiple websites with different cryptocurrency wallet addresses.

Bibliography

[1] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. URL https://bitcoin.org/bitcoin.pdf.

[2] coinhive. Coinhive – Monero JavaScript Mining, 2018. URL https://coinhive.com/.

[3] L. Kelion. Starbucks cafe's wi-fi made computers mine crypto-currency - BBC News, 2017. URL http://www.bbc.com/news/technology-42338754.

[4] S. Liao. Showtime websites secretly mined user CPU for cryptocurrency - The Verge, 2017. URL https://www.theverge.com/2017/9/26/16367620/showtime-cpu-cryptocurrency-monero-coinhive.

[5] B. Fung. Hackers have turned Politifact's website into a trap for your PC - The Washington Post, 2017. URL https://www.washingtonpost.com/news/the-switch/wp/2017/10/13/hackers-have-turned-politifacts-website-into-a-trap-for-your-pc/.

[6] Brave - Secure, Fast & Private Web Browser with Adblocker. URL https://brave.com/.

[7] Brave Software. Basic Attention Token (BAT): Blockchain Based Digital Advertising. 2018. URL https://basicattentiontoken.org/BasicAttentionTokenWhitePaper-4.pdf.

[8] S. Eskandari, A. Leoutsarakos, T. Mursch, and J. Clark. A first look at browser-based Cryptojacking. 2018. URL http://arxiv.org/abs/1803.02887.

[9] Salon. Salon: in-depth news, politics, business, technology and culture, 2018. URL https://www.salon.com/.

[10] Adi Robertson. Salon asks ad-blocking users to opt into cryptocurrency mining instead - The Verge, 2018. URL https://www.theverge.com/2018/2/13/17008158/salon-suppress-ads-cryptocurrency-mining-coinhive-monero-beta-testing.

[11] Litecoin - Open source P2P digital currency. URL https://litecoin.org/.

[12] G. Wood. Ethereum: a secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, pages 1–32, 2014. ISSN 1098-6596. doi: 10.1017/CBO9781107415324.004.

[13] Ethereum Foundation. Ethereum Project, 2015. URL `https://www.ethereum.org/`.

[14] Cryptokitties. CryptoKitties - Collect and breed digital cats, 2018. URL `https://www.cryptokitties.co/https://www.cryptokitties.co/about`.

[15] M. E. Peck. Blockchains: How They Work and Why They Will Change the World - IEEE Spectrum, 2017. URL `https://spectrum.ieee.org/computing/networks/blockchains-how-they-work-and-why-theyll-change-the-world`.

[16] M. Bartoletti and L. Pompianu. An analysis of bitcoin OP_RETURN metadata. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10323 LNCS:218–230, 2017.

[17] S. Cohen and A. Zohar. Database Perspectives on Blockchains. pages 1–25, 2018. doi: 10.4230/LIPIcs. URL `http://arxiv.org/abs/1803.06015`.

[18] D. Martens and W. Maalej. ReviewChain: Untampered Product Reviews on the Blockchain. pages 1–4, 2018. URL `http://arxiv.org/abs/1803.01661`.

[19] Nuances Between Permissionless and Permissioned Blockchains. URL `https://medium.com/@akadiyala/nuances-between-permissionless-and-permissioned-blockchains`.

[20] M. Hearn. Corda: A distributed ledger. *Whitepaper*, pages 1–56, 2016.

[21] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. 2018. doi: 10.1145/3190508.3190538. URL `http://arxiv.org/abs/1801.10228`.

[22] Home - Hyperledger. URL `https://www.hyperledger.org/`.

[23] r3.com. URL `https://www.r3.com/`.

[24] C. Gusson. 126 Banks-Strong Brazilian Group Febraban Reveals Blockchain Tests, 2018. URL `https://www.ccn.com/febraban-an-organization-that-brings-together-120-banks-in-brazil/reveals-tests-in-blockchain/`.

[25] A. Antonovici. IBM's Batavia Successfully Tested for Live Transactions, Audi Involved - Cryptovest, 2018. URL https://cryptovest.com/news/ibms-batavia-successfully-tested-for-live-transactions-audi-involved/.

[26] W. Zhao. Huawei Unveils Hyperledger-Powered Blockchain Service Platform - CoinDesk, 2018. URL https://www.coindesk.com/huawei-unveils-hyperledger-based-blockchain-service-platform/.

[27] I. Allison. Japan Net Bank tests blockchain with mijin and Hyperledger, 2018. URL https://www.ibtimes.co.uk/japan-net-bank-tests-blockchain-mijin-hyperledger-1659181.

[28] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982. URL http://portal.acm.org/citation.cfm?doid=357172.357176.

[29] C. Cachin and M. Vukolić. Blockchain Consensus Protocols in the Wild. 2017. ISSN 18688969. doi: 10.4230/LIPIcs.DISC.2017.1. URL http://arxiv.org/abs/1707.01873.

[30] Cardano - Home of the Ada cryptocurrency and technological platform. URL https://www.cardano.org/en/home/.

[31] W. Li, S. Andreina, J. M. Bohli, and G. Karame. Securing proof-of-stake blockchain protocols. *Lecture Notes in Computer Science*, 10436 LNCS:297–315, 2017.

[32] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. *Lecture Notes in Computer Science*, 10401 LNCS:357–388, 2017.

[33] V. Buterin and V. Griffith. Casper the Friendly Finality Gadget. pages 1–10, 2017. URL http://arxiv.org/abs/1710.09437.

[34] H. Partz. Ethereum Devs Publish Upgrade Proposal To Move Network Away From Mining-Related Issues, 2018. URL https://cointelegraph.com/news/ethereum-devs-publish/upgrade-proposal-to-move-network-away-from/mining-related-issues.

[35] M. Castro and B. Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems*, 20(4):398–461, 2002. URL http://portal.acm.org/citation.cfm?doid=571637.571640.

[36] A. Bessani, J. Sousa, and E. E. Alchieri. State machine replication for the masses with BFT-SMART. *Proceedings - 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2014*, pages 355–362, 2014. doi: 10.1109/DSN.2014.43.

[37] Y. Sompolinsky and A. Zohar. Secure high-rate transaction processing in bitcoin. In *Lecture Notes in Computer Science*, volume 8975, pages 507–527, 2015.

[38] Lombrozo Eric, Lau Johnson, and Wuille Pieter. Segregated Witness (Consensus layer) proposal - Github. URL `https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki`.

[39] J. Poon and T. Dryja. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. *Technical Report (draft)*, page 59, 2016. URL `https://lightning.network/lightning-network-paper.pdf`.

[40] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena. A Secure Sharding Protocol For Open Blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16*, 2016.

[41] Stratum mining. All about cryptocurrency - Bitcoin Wiki. URL `https://en.bitcoinwiki.org/wiki/Stratum`.

[42] Getwork RPC Method - Bitcoin Wiki. URL `https://en.bitcoin.it/wiki/Getwork`.

[43] R. Recabarren and B. Carbunar. Hardening Stratum, the Bitcoin Pool Mining Protocol. pages 1–18, 2017.

[44] Difference between ASIC, GPU, and CPU mining — CoinTopper. URL `https://cointopper.com/guides/difference-between-asic-gpu-and-cpu-mining`.

[45] GPU & CPU benchmarks for Monero mining. URL `https://monerobenchmarks.info/`.

[46] N. Van Saberhagen. Monero: CryptoNote v 2.0. 2013. URL `https://cryptonote.org/whitepaper.pdf`.

[47] CryptoNote Technology. URL `https://cryptonote.org/inside.php#equal-proof-of-work`.

[48] CryptoNoter. CryptoNoter Github Repository, 2018. URL `https://github.com/cryptonoter`.

[49] XMRIG. Stratum Protocol Extension XMRIG, 2018. URL `https://github.com/xmrig/xmrig-proxy/blob/dev/doc/STRATUM`.

[50] WhatToMine. Monero (XMR) Mining Profit Calculator - WhatToMine, 2018. URL https://whattomine.com/coins/101-xmr-cryptonightv8.

[51] CryptoLoot. CryptoLoot - Profit Calculator, 2018. URL https://crypto-loot.com/.