Web Monetization Using Crypto-Currencies

Natalino António Bernardino Cordeiro
Instituto Superior Técnico, Universidade de Lisboa

***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 an 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 this 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.

I. 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 consumers 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.

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 Chaums 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.

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.

A. 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. 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.

II. Background and Related Work

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.

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.

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 [3], 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 [4] was reported as the first website with a large audience to adopt this alternative as an option for users of ad-blocking software [5].

A. Blockchain

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. 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.

B. 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.

III. Conceptual Design

Considering the web-mining services currently present in the market, 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.

A. 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 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 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

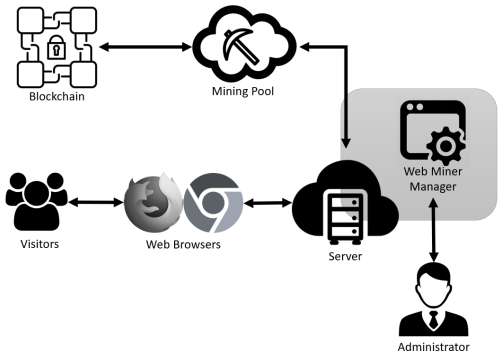

Fig. 1: Entities that participate in the web mining process

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* 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, 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*.

B. Used Technologies

The developed project is based on a series of protocols and algorithms that are already present in similar systems.

1) Stratum Protocol: Stratum [6] 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 [7] 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 $\backslash\mathbf{r}\backslash\mathbf{n}$. 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 $\backslash\mathbf{n}$. 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*.

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.

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.

2) CryptoNote: CryptoNote [8] 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* 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.

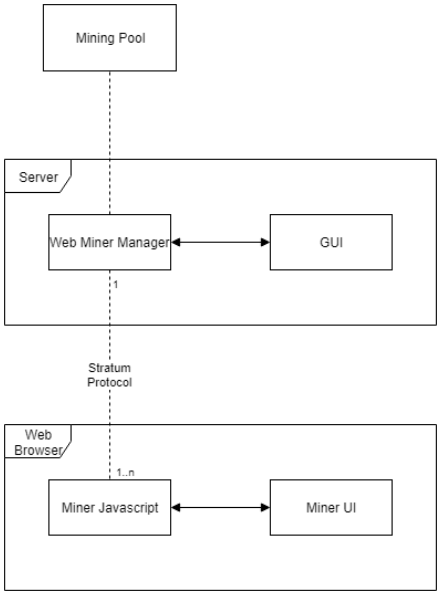

Fig. 2: Overview of the component structure

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.

C. Component Overview

The component structure of the developed project can be seen in Figure 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.

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 webpage 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), 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 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.

D. 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.

IV. Implementation

The implementation of the project can be divided in two major sides, the *Web Miner Manager* that works on the server side, and the in-browser miner that works on the client side.

A. 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,

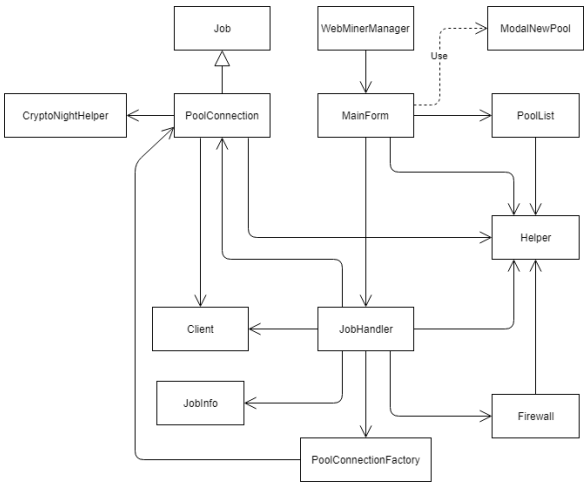

Fig. 3: Web Miner Manager's simplified class diagram

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 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 is built on top of CryptoNoter's [9] implementation of the *Stratum Protocol*, 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.

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.

2) Stratum Protocol Extensions: This web-mining system was developed to be compatible with CryptoNight and CryptoNight-lite cryptocurrencies but, 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 [10], 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 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.

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 *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.

B. In-Browser Miner Implementation

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.

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, function *startMining()* is executed.

This function simply verifies if all the conditions necessary for the browser to mine are met, and if so, it starts the mining process.

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.

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.

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.

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, 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.

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.

Since both these elements are built using only CSS and HTML code, they are easily configurable to fit a wide range of websites.

V. Experimental Evaluation

Considering 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, 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.

A. 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.

Figure 4 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 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.

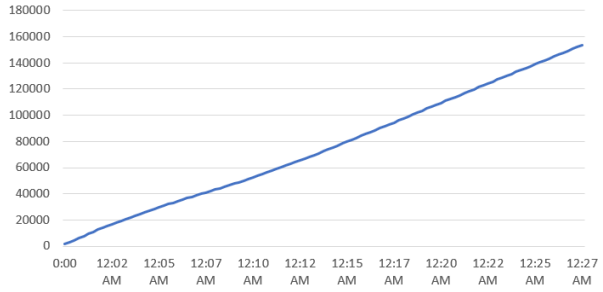

Fig. 4: Hashes calculated over time by 1 client running on *Machine 1* at 100%

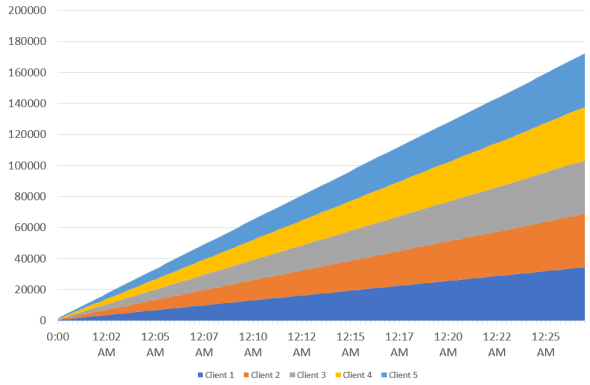

Fig. 5: Hashes calculated over time by 5 clients running on *Machine 1* at 100%

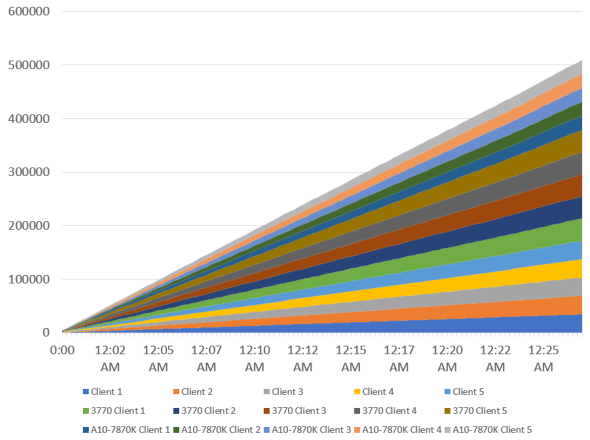

Fig. 6: Hashes calculated by 15 clients running on 3 machines at 100%

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 6.

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.

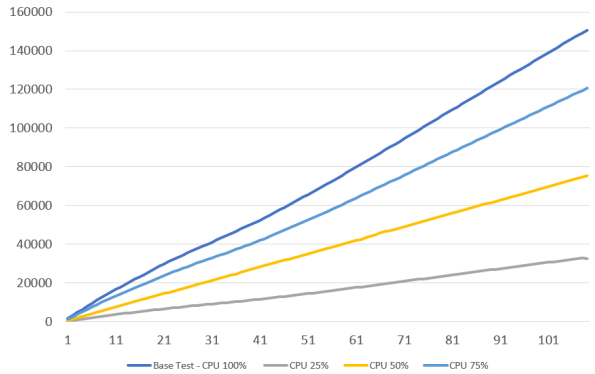

Fig. 7: Hashes calculated over time by 1 clients running on *Machine 1*

B. 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 4 was repeated, but with the slider bar at 25%, 50% and 75%. The results of this test can be observed in Figure 7.

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.

C. Revenue and Costs

This section presents an estimation of how much revenue the developed solution could generate for websites with user bases of various sizes, and compares those results with the estimated revenue of an ad-based model.

For 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* [11], with a valuation of each *Monero* token at 52.49$, at the

TABLE I: 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

time of writing this document. Any costs associated with the conversion of cryptocurrencies to fiat currency are ignored.

The test illustrated by Figure 4 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 I. This first example shows that any website that averages 1 connected user at the time, would have an estimated profit each month of $-\$39.61$.

D. 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 services will proliferate in the future. As such, considering our estimations, 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

TABLE II: 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

$$CPCProfit = V * clickrate * payment$$

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 II, 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.

E. Other Web Miners

Unfortunately *CoinHive* does not make available a profit estimation calculator for their service, but *CryptoLoot*, a competitor to *CoinHive*, does [12]. Using that calculator it is possible to estimate how the results shown in Table II 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 8.

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 adhering 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.

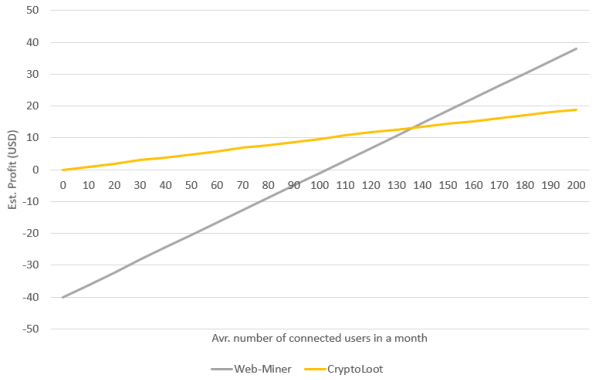

Fig. 8: Estimated profitability of the solution vs CryptoLoot

VI. Conclusion

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.

A. 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.

B. 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;
- 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.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[2] coinhive, "Coinhive Monero JavaScript Mining," 2018. [Online]. Available: https://coinhive.com/
[3] S. Eskandari, A. Leoutsarakos, T. Mursch, and J. Clark, "A first look at browser-based Cryptojacking," 2018. [Online]. Available: http://arxiv.org/abs/1803.02887
[4] Salon, "Salon: in-depth news, politics, business, technology and culture," 2018. [Online]. Available: https://www.salon.com/
[5] Adi Robertson, "Salon asks ad-blocking users to opt into cryptocurrency mining instead - The Verge," 2018. [Online]. Available: https://www.theverge.com/2018/2/13/17008158/salon-suppress-ads-cryptocurrency-mining-coinhive-monero-beta-testing
[6] "Stratum mining. All about cryptocurrency - Bitcoin Wiki." [Online]. Available: https://en.bitcoinwiki.org/wiki/Stratum
[7] "Getwork RPC Method - Bitcoin Wiki." [Online]. Available: https://en.bitcoin.it/wiki/Getwork
[8] N. Van Saberhagen, "Monero: CryptoNote v 2.0," 2013. [Online]. Available: https://cryptonote.org/whitepaper.pdf
[9] CryptoNoter, "CryptoNoter Github Repository," 2018. [Online]. Available: https://github.com/cryptonoter
[10] XMRIG, "Stratum Protocol Extension XMRIG," 2018. [Online]. Available: https://github.com/xmrig/xmrig-proxy/blob/dev/doc/STRATUM
[11] WhatToMine, "Monero (XMR) Mining Profit Calculator - WhatToMine," 2018. [Online]. Available: https://whattomine.com/coins/101-xmr-cryptonightv8
[12] CryptoLoot, "CryptoLoot - Profit Calculator," 2018. [Online]. Available: https://crypto-loot.com/