The Hidden Layers of Blockchains: Technical Nuances and their Unforeseen Consequences

Shayan Eskandari

A Thesis in
The Concordia Institute for Information Systems Engineering (CIISE)

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy
(Information and Systems Engineering)
at
Concordia University
Montréal, Québec, Canada

January 2024

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Shayan Eskandari**

Entitled: **The Hidden Layers of Blockchains: Technical Nuances and their Unforeseen Consequences**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Information and Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________ Chair
TBD

______________________ External Examiner
TBD

______________________ Examiner
Amr Youssef

______________________ Examiner
M. Mannan

______________________ Arms Length Examiner
Ketra Schmitt

______________________ Supervisor
Jeremy Clark

Approved by ______________________
Chun Wang, Graduate Program Director (CIISE)

01 January 2024 ______________________
Mourad Debbabi, Dean (GCS)

Abstract

Name: **Shayan Eskandari**

Title: **The Hidden Layers of Blockchains: Technical Nuances and their Unforeseen Consequences**

Hello. No more than 250 words.

Acknowledgments

Hello.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Motivation. In the course of my academic journey, my professional experiences have played a pivotal role in shaping and enriching my research focus. Here is a brief overview of my double life and the experiences that have led me to this dissertation:

Starting from my role as the `Blockchain Engineer` at *Bitaccess* (a Bitcoin ATM company, later *NASDAQ: BTM*), I learned that Bitcoin is used by some people in a daily basis, most of whom were experiencing some level of friction with the current financial system and its limitations. It should be noted that the user base was not exclusive to the *unbanked*, many others were technologists, enthusiasts or abroad workers remittances to their families or friends in other countries. This lead me to explore the technical nuances of different wallets, payment gateways, and key management approaches in my master's degree [114], both for the users [118] and the merchants willing to accept Bitcoin as a form of payment [119].

It was during this role that I also noticed that the public perception of cryptocurrencies as criminal money had solidified. This perception is not meritless, as there are many cases of fraud and theft in the blockchain space. Due to the *pseudonymity*[1] and the irreversible nature of the cryptocurrency transactions, we see this technology used for a range of criminal activities, specially for (online) scams and ransomware [252, 84]. However, the blockchain technology is not inherently criminal. In fact, the technology has the potential to reduce the fraud rate by providing a transparent, global, and immutable ledger. We noticed that as a service provider, we are able to prevent some of the fraudulent transactions by early screening of the recipient address, and contacting the user for verification. This is only feasible due to the public nature of the blockchain and early detection of the scammers' bitcoin addresses.

Later on, my role as a `Security Engineer and Smart Contract Auditor` at *ConsenSys Diligence* provided me with valuable insights into the world of smart contract security. This role was instrumental in deepening my understanding of the technical complexities and composability within the blockchain ecosystem and Decentralized Finance (DeFi). Engaging closely with the nuances of the technology and different implementations, I developed a comprehensive perspective on its operational frameworks and security paradigms. This has enabled me to identify many false technical assumptions that are not obvious to the developers and users alike [117, 115].

As the `Chief Technology Officer (CTO)` at *Ether Capital (ETHC.NE)*, a publicly traded entity with the majority of holdings in cryptoassets, I was exposed to

[1] Pseudonymity means even though the true identity behind the address is not known, but the address itself is an entity (in which, one might leak information about the real identity behind the entity).

the practicalities of integrating cryptoassets into the established financial ecosystem. This experience was transformative, offering me a first-hand experience to the dialogue, strategies, and implementation processes by real-world companies as they navigate the domain of cryptoassets. It was during this time that I engaged extensively with various financial institutions and auditors, particularly in the context of annual audits and quarterly financial statements. These interactions revealed a significant disconnect between traditional auditing methods and the nuanced requirements of cryptoassets custody and ownership. I observed the overcomplicated and redundant processes that failed to address the core needs of companies and stakeholders in the realm of cryptocurrencies.

Through this thesis, I seek to shed some light on some of these knowledge gaps, and discuss the potential ethical issues and technical solutions to narrow these gaps. A few examples: In chapter 3, we discuss the ethical questions surrounding *cryptojacking* as a replacement for the online advertisements, but due to the policy vacuum it fell victim to the bad actors. In chapter 4, we discuss how some applications and solutions to "democratize" the profit from blockchain front-running attacks, fall in gray areas of ethics and regulations. In chapter 6, we discuss the challenges of auditing cryptoassets and the disconnect between the traditional auditing methods and the technical expertise required to verify a financial statement involving cryptoassets. There are many more examples throughout this thesis that we discuss the ethical and technical nuances of the blockchain technology and relevant challenges.

1.1 Contributions and Outline

In this dissertation, we discuss the technical nuances of the blockchain technology and its applications, conduct research on their unforeseen consequences that enabled bad actors to profit from unethical use of the technology. The *pseudonymity*, as well as, irreversible, transparent, and permissionless aspect of the blockchain technology, enables more actors to participate in the misdeeds, which previously was only limited to a few actors (e.g. brokers in the traditional financial systems).

In **Chapter 2**, we offer a background and an introduction to the essential concepts necessary for comprehending this dissertation. Each chapter also includes a background section to extend the necessary concepts for the chapter.

In **Chapter 3**, we present the first academic study of cryptojacking, covering its fast-paced growth and the ethical questions surrounding this use of the blockchain technology. *In-browser Cryptojacking* is the use of cryptocurrency mining technology to replace common online advertisement revenue mechanisms. We examine the trend towards in-browser mining of cryptocurrencies; in particular, the mining of Monero through Coinhive and similar services. This technology was introduced as a novel mean for monetizing web content, as a way to replace all the intermediaries in the typical means of advertisement and/or paywalls. In this model, a user visiting a website will seamlessly download a JavaScript code that executes in her browser, mines a cryptocurrency—typically without her consent or knowledge—and pays out the seigniorage to the website. Websites may consciously employ this as an alternative or to supplement advertisement revenue, may offer premium content in exchange for mining, or may be unwittingly serving the code as a result of a breach (in which case the seigniorage is collected by the hacker). The cryptocurrency Monero is preferred seemingly for its unfriendliness to large-scale ASIC mining that would drive browser-based efforts out of the market, as well as for its purported privacy features. We survey this landscape, conduct some measurements to establish its prevalence and profitability, outline an ethical framework for considering whether it should be classified as an attack or business opportunity, and make suggestions for the detection, mitigation and/or prevention of browser-based mining for non-consenting users.

In **Chapter 4**, we study the front-running attacks on blockchain applications, provide a taxonomy of these attacks, and analyze the common mitigation methods. We consider front-running to be a course of action where an entity benefits from prior access to privileged market information about upcoming transactions and trades. Front-running has been an issue in financial instrument markets since the 1970s. With the advent of the blockchain technology, front-running has resurfaced in new forms we explore here, instigated by blockchain's decentralized and transparent nature. Due to the public and transparent aspect of blockchains, any actor can act as the privileged actor in the information flow and abuse this early access to information to profit from it. In this chapter, we draw from a scattered body of knowledge and instances of front-running across the top 25 most active decentralized applications (DApps) deployed on Ethereum blockchain. Additionally, we carry out a detailed analysis of *Status.im* initial coin offering (ICO) and show evidence of abnormal miner's behavior indicative of front-running token purchases. Finally, we map the proposed solutions to front-running into useful categories. Lastly, we discuss the ethical and legal issues that are left unanswered so far.

In **Chapter 5**, we study the oracle implementations on the blockchain, and the attacks on blockchain applications that use oracles. *Oracles* are the gateways between the smart contracts and the real-world information. One fundamental limitation of blockchain-based smart contracts is that they execute in a closed environment. Thus, they only have access to data and functionality that is already on the blockchain, or is fed into the blockchain. Any interactions with the real world need to be mediated by a bridge service, which is called an *oracle*. As decentralized applications mature, oracles are playing an increasingly prominent role. With their evolution comes more attacks, necessitating greater attention to their trust model. We dissect the design alternatives for oracles, showcase attacks, and discuss attack mitigation strategies.

In **Chapter 6**, we study the auditing of blockchain-based assets and the systematic challenges a company (and the auditing firms) might have to properly audit their crypto-assets. Auditing firms are hesitant to accept mandates from companies that hold a significant amount of cryptoassets, primarily because the blockchain sector introduces novel, technically sophisticated, and risky propositions that auditors are unequipped to handle. Abrupt recusals by auditors operating in this sector have led to several enterprises being placed on cease trade by securities regulators for failure to produce audited financial statements on time, thus impeding these companies from raising capital and bringing new investments to fund innovation in this space. In this chapter we critically analyze the purported roadblocks to auditing blockchain firms and map them to traditional auditing practices. We discuss four different case studies and the challenges those solutions bring forth. We urge auditors to reconsider their resistance to the blockchain sector by demonstrating that providing an audit opinion is challenging but not insurmountable.

Lastly, in **Chapter 7**, we provide some concluding remarks and future research prospects.

1.2 Additional Publications

In this section, I present a brief summary of publications I wrote during my doctoral studies that are not directly related to the main theme of this dissertation. These publications are the result of collaborations with other researchers and are not included in the main body of this dissertation.

- **The Middleman is Dead, Long Live the Middleman: The "trust factor" and the psycho-social implications of blockchain** [144][2]: Blockchain is widely regarded as a breakthrough innovation that may have a profound impact on the economy and society, of a magnitude comparable to the effects of the introduction of the Internet itself. In essence, a blockchain is a decentralized peer-to-peer network with no central authority figure, which adds information to the distributed database by collectively validating the accuracy of data. Since

[2]Published in *Frontiers in Blockchain Journal. 2019, 2, 20*

each node of the network participates in the review and confirmation of the new information before being accepted, the need for a trustworthy intermediary is eliminated. However, as trust plays an essential role in affecting decisions when transacting with one another, it is important to understand which implications the decentralized nature of blockchain may have on individuals' sense of trust. In this contribution, we argue that the adoption of blockchain is not only a technological, but foremostly a psychological challenge, which crucially depends on the possibility of creating a trust management approach that matches the underlying distributed communication system. We first describe the decentralization technologies and possibilities they hold for the near future. Next, we discuss the psycho-social implications of the introduction of decentralized processes of trust, examining some potential scenarios, and outline a research agenda.

- **Resolving the Multiple Withdrawal Attack on ERC20 Tokens** [274][3]: Custom tokens are an integral component of decentralized applications (DApps) deployed on Ethereum and other blockchain platforms. For Ethereum, the ERC20 standard is a widely used token interface and is interoperable with many existing DApps, user interface platforms, and popular web applications (e.g., exchange services). An ERC20 security issue, known as the multiple withdrawal attack, was raised on GitHub and has been open since October 2017. The issue concerns ERC20's defined method approve() which was envisioned as a way for token holders to give permission for other users and DApps to withdraw a

[3]Published in *European symposium on security and privacy workshops (EuroS&PW). IEEE, 2019*

capped number of tokens. The security issue arises when a token holder wants to adjust the amount of approved tokens from N to M (this could be an increase or decrease). If malicious, a user or DApp who is approved for N tokens can front-run the adjustment transaction to first withdraw N tokens, then allow the approval to be confirmed, and withdraw an additional M tokens. In this paper, we evaluate 10 proposed mitigations for these issues and find that no solution is fully satisfactory. We then propose 2 new solutions that mitigate the attack, one of which fully fulfills constraints of the standard, and the second one shows a general limitation in addressing this issue from ERC20's approve method.

Chapter 2

Background

In this chapter, we go over the building blocks for blockchain technology and other concepts required for understanding this dissertation.

2.1 Blockchain

A technical challenge in developing a decentralized network has been establishing a consensus mechanism (See section 2.1.1) that operates without centralized trust among participants. "Trust" here refers to the reliance on a central authority to validate transactions, a notion Bitcoin managed to eliminate. Implementing an online ledger on a single computer is straightforward, involving a database with public read access and restricted write access controlled by a central authority. This setup inherently trusts the central party managing the server. However, as soon as you begin to have more computers in the decentralized network, the question arises, how to keep updating the database and enforce honesty amongst the network participants? Prior to Bitcoin, attempts to decentralize this model were made starting in the 1980's [237], but it wasn't until Nakamoto's work in 2008 [234] that a viable solution emerged, leveraging a network of nodes incentivized to maintain and validate a public ledger without a central authority. A decentralized network here means that no single entity controls the network, and the network is open to anyone to join or leave at any time.

A (full) node in the Bitcoin network, is a computer that maintains a copy of the blockchain and is verifying transactions and blocks. Nodes that are configured to create new blocks, called a miner, are rewarded with new cryptocurrency (bitcoin) for their efforts in transaction verification and block formation (further explained in 2.1.1.1). Transactions, essentially value transfers, are the fundamental components of the blockchain. The blockchain is a list of such transactions grouped into sequential blocks (explained in more details in 5.5.2.2). The order within these blocks is critical, indicating the sequence of transactions across the network (See Chapter 4).

Public blockchains are a promising underlying technology for many applications. Their aim of providing decentralization, transparency, and immutability correlates with the security goals of many use-cases. However, blockchains are also slow, expensive, and have practical limits to their functionality. I predict they can and will replace many intermediary entities we know of today. There is a gap in the mental model of what people (industry and academia) think blockchain provides and what it actually does [280]. Even developers need to change many assumptions given the pub-

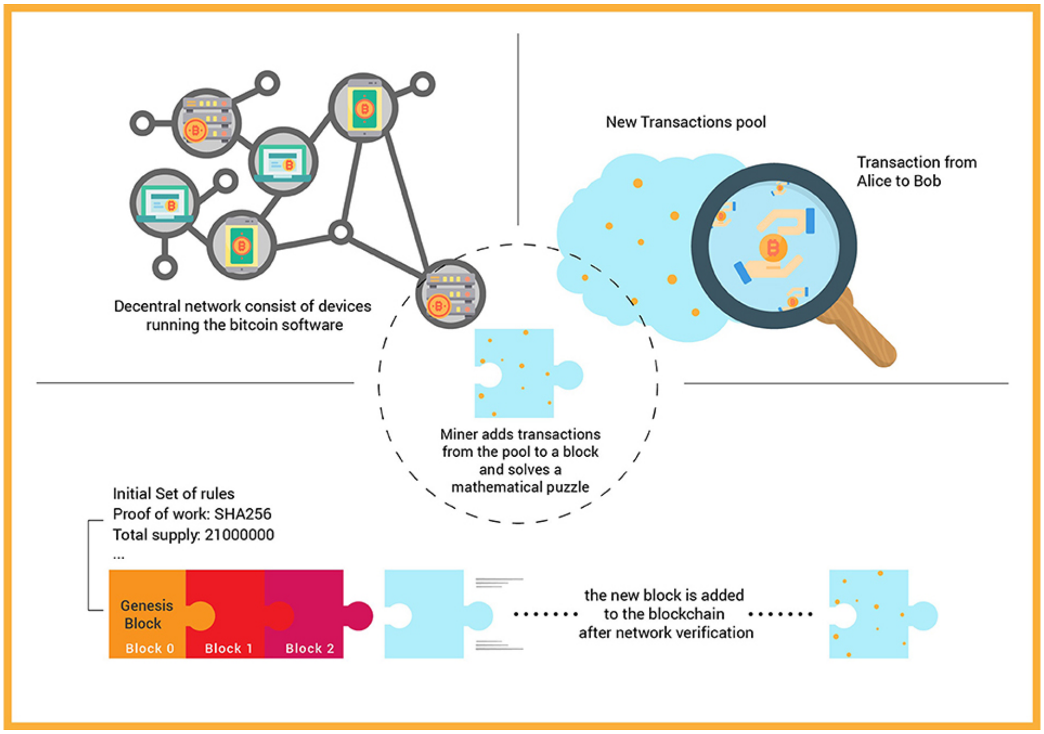

Figure 2.1: The building blocks of blockchain technology

lic aspect of how a blockchain works, and system designers have to rethink the data flow within their applications. We hope to illustrate this throughout the dissertation.

This technology has the potential of many interesting applications, from financial use cases (*e.g.,* Bitcoin [234]), asset trading and markets [70, 290], insurance and futures [212, 229], tamper-resistant record storage [35], and online voting [216, 13]. Bitcoin, started in 2009, was the first application of blockchain technology and since then the concept of decentralized ledger has grown to many other applications that just a ledger holding transaction data.

Bitcoin pioneered the use of a decentralized ledger for tracking cryptocurrency transactions. However, its limitations prompted the development of other blockchain technologies seeking to enhance its original model. Ethereum, introduced by Vitalik

Buterin, expanded on Bitcoin's capabilities by incorporating smart contracts, allowing for complex, programmable transactions beyond simple value transfers [55]. This added functionality, however, comes at the cost of speed and efficiency, with Ethereum facing scalability challenges. To address these, developers are exploring solutions such as layer 2 protocols [72, 249, 172], which operate on top of the Ethereum blockchain to increase transaction speeds and reduce costs.

In parallel, privacy-focused cryptocurrencies like Monero [226] and Zcash [165] have been developed to address another limitation of Bitcoin: privacy. These platforms implement advanced cryptographic techniques to offer enhanced privacy and anonymity for users, making transaction details and participant identities obscure [326, 220]. Monero, for example, uses ring signatures and stealth addresses to protect user privacy [243], distinguishing itself from Bitcoin's transparent transaction ledger. It should be noted that these privacy features are not perfect yet and have been found vulnerable in the past [190, 222].

2.1.1 Consensus Mechanism

A consensus mechanism is a fundamental protocol within blockchain technology that enables network participants, often referred to as validators[1], to agree on the state of the ledger, despite the absence of anyone coordinating their actions or them trusting each other fully. It is the bedrock of blockchain's reliability and security, ensuring that each participant has a consistent view of the transaction record (final state of

[1] In this section, we use *miners* and *validators* interchangeably for nodes that validate transactions and participate in block creation.

the blockchain). This mechanism is crucial in decentralized systems like Bitcoin and Ethereum, where no central authority confirms the validity of transactions, or dictates who can play the role of a validator in the system.

In essence, a consensus mechanism involves a series of processes and rules that nodes follow to validate new transactions, add them to the blockchain, and achieve agreement on the current state of the ledger. If all nodes are malicious, consensus is impossible; however protocols assume that at all times, a threshold (*e.g.,* 50%; although the exact number and nature varies) of honest nodes will follow these processes and rules. These processes are designed to prevent fraudulent activities and to mitigate the influence of any single entity over the network. The most widely recognized consensus mechanisms are based on proof of work (PoW) and proof of stake (PoS).

2.1.1.1 Proof of Work (PoW)

Employed initially by Bitcoin and Ethereum, PoW requires nodes, called *miners*, to solve cryptographic puzzles. Miners must solve the puzzle before being able to add a new block of transactions to the blockchain. This method, while robust in maintaining network security, is often criticized for its high energy consumption. In order to understand PoW a few cryptography primitives and other building blocks need to be explained.

Hash Functions. A hash function is a deterministic algorithm that takes an input (or 'message') and returns a fixed-size string of bytes, typically a digest that appears random. The output, known as the *hash value* or the hash, is unique to each unique input: even a minor change in the input results in a significantly different output. This property, called the avalanche effect [136], is crucial for security. Another important property of hash functions in the context of blockchain is *collision resistance*, which ensures that it is highly improbable for two different inputs to produce the same output hash. This attribute is vital for maintaining the integrity of the blockchain, as it helps prevent tampering by ensuring that each block can be uniquely identified and verified through its hash. Hash functions are used for various purposes in blockchain, including generating addresses, creating digital fingerprints of data, and linking each block to the previous blocks in the blockchain securely. The nature of hash functions makes them one-way operations—while it's feasible to generate a hash from a given piece of data, it's computationally infeasible to reconstruct the original data from the hash. This one-way property, coupled with collision resistance, is crucial for blockchain security and digital signatures, as it prevents malicious actors from tampering with the data by making it exceedingly difficult to alter the data without detection.

Digital Signature. Before delving into the operations of digital signatures, it's crucial to understand the context of Public Key Infrastructure (PKI) [4], which underpins the security of digital signatures. PKI is a framework that describes encryption keys, including the creation, distribution, and verification of public and private keys. In this system, each user has a pair of keys: a public key and a private key. The public key is derived from the private key through cryptographic hash functions, ensuring that both keys are interconnected. The private key must be kept secure by the owner, as it is used to create digital signatures and blockchain transactions. Furthermore, the public key can be shared with anyone and is used to verify the authenticity of the message signed with the corresponding private key. If a private key is compromised or stolen, it could lead to impersonation of the identity behind the pubkey, unauthorized transactions, and potentially significant loss of assets, underscoring the importance of robust internal controls to safeguard private keys (See Section 6.3.4).

Digital signatures are a cryptographic tool used to ensure authenticity and integrity in digital communications and transactions. They function as a digital equivalent of a handwritten signature or stamped seal, providing enhanced security. This mechanism is pivotal for securing transactions on a blockchain. A digital signature process encompasses two main operations: signing and verifying. The signing operation involves generating a signature where an individual utilizes their private key to sign a message (or transaction) using a specific signature algorithm. This signature, once generated, is appended to the message and forwarded to the recipient. The verification process is initiated by the recipient, who employs the corresponding public key to apply a verification algorithm to the signature. A valid signature confirms that the message remains unchanged and authenticates that it was signed by the private key holder. This verification process is integral to blockchain transactions, safeguarding against tampering and ensuring transactions are legitimately conducted by the true owners of the digital assets behind the corresponding public key.

Following the creation and signing of a transaction, it enters the network and is propagated to other nodes through a process known as gossiping. This propagation ensures that the transaction reaches validators, who collect and store these transactions in a temporary storage area called the mempool. The mempool acts as a waiting room for transactions before they are included in a new block. It's important to note that the order in which transactions are selected from the mempool by validators is arbitrary and can vary from one validator to another, depending on factors such as transaction fees offered. Validators then compile these transactions into a block, applying their computational resources to solve the cryptographic challenge associated with block creation. We delve deeper into these concepts in the next sections, as well as, in sections 5.5.2 and 4.2.3.

Mining. During the mining process, miners engage in a competitive effort to solve a cryptographic puzzle, necessitating the discovery of a hash that aligns with a specific condition imposed by the network—typically an output value that falls below a designated target (*difficulty*). This process requires the incorporation of data from the block header, including a reference to the preceding block's hash to maintain the blockchain's integrity and continuity, along with a collection of signed transactions from the mempool. Through iterative hashing of this composite data with a variable nonce, miners compete to generate a valid hash for the block that satisfies the network's difficulty standards.

Upon successfully solving the puzzle, a miner earns the right to append the newly formed block to the blockchain. The competitive nature of mining introduces the potential for simultaneous discoveries, leading to temporary forks in the blockchain. Such forks are resolved over time as the network reaches consensus on which block to extend, a process that may involve reorganizations (re-orgs) of the blockchain to select the most valid chain. The winning block consolidates all verified transactions, ensuring their authenticity and validity via digital signatures linked to their respective private keys. The miner whose block gains acceptance by the network is rewarded with newly minted cryptocurrency (like Bitcoin or Ether) and transaction fees from the transactions in the block they mined, incentivizing the contribution of computational resources. The history of mining ecosystem is by itself an interesting story which some of it is discussed in 3.3.1.

Other than incidental temporary forks, intentional forks within the blockchain can manifest as either hard forks or soft forks, each with distinct implications for the network. Hard forks introduce backward-incompatible changes that require all nodes to upgrade to the new protocol version, whereas soft forks entail backward-compatible modifications that do not necessitate universal adoption by all nodes. This distinction becomes particularly relevant in discussions surrounding major network updates and the emergence of new cryptocurrencies, such as Bitcoin Cash from Bitcoin [329].

The concept of transaction confirmations further enriches the mining narrative. A transaction without any confirmations (0-confirmation) is considered unverified, while one that is included in a block (1-confirmation) has been verified by the network. Additional confirmations occur as subsequent blocks are added, increasing the transaction's security. This is particularly important in the context of double-spending attacks, where a malicious actor attempts to spend the same cryptocurrency twice. The more confirmations a transaction has, the more secure it is against such attacks.

A critical security concern in the mining process is the *51% attack*, where an entity gains control of the majority of the network's mining power. This dominance allows the attacker to potentially double-spend coins and prevent the confirmation of new transactions (censorship), undermining the blockchain's security and trustworthiness.

Mining is not just about creating new cryptocurrency units; it's the backbone of maintaining the decentralization, security, and integrity of the blockchain. By utilizing hash functions, it ensures that each block is securely linked to its predecessor, creating an immutable ledger. The reliance on digital signatures within each transaction enhances the security and authenticity of the transactions recorded in each block. The decentralized nature of mining, where multiple nodes compete to solve the puzzle, ensures that no single entity can control the network. This mechanism is the foundation of blockchain security.

Mining Pools. Mining pools exist particularly in networks that utilize the Proof of Work (PoW) consensus mechanism. They consist of groups of miners who combine their computational resources to increase their collective probability of successfully mining a block and receiving the rewards. Individual miners, especially those with limited hardware capabilities, might find it challenging to compete against more powerful, resource-rich miners. By joining a mining pool, these smaller miners effectively pool their hashing power, increasing their chances of solving the cryptographic puzzles required to add a block to the blockchain. When a pool successfully mines a block, the reward is distributed amongst its members in proportion to the computational power each contributed. This arrangement makes mining more accessible and potentially more profitable for individual miners, as it offers a more consistent and predictable return compared to solo mining, where rewards can be sporadic and significantly less frequent. Mining pools are instrumental in maintaining a degree of democratization in the mining process, ensuring that smaller players can still contribute to and benefit from the blockchain network. That being said, if a mining pool becomes too large, it can potentially control the network, leading to centralization and security concerns. We discuss more relevant details on mining pools in Chapter 3.3.1.2.

2.1.1.2 Proof of Stake (PoS).

Proof of Stake is a system that relies on validators who "stake" their cryptocurrency holdings as collateral to participate in the process of managing the blockchain. Unlike Proof of Work (PoW), which requires energy-intensive computational work, PoS selects validators based on the amount of their staked cryptocurrency, creating a more energy-efficient process.

In PoS, the likelihood of a validator being chosen to propose or validate a block correlates with the size of their stake. Validators lock up a certain amount of cryptocurrency (32 ETH in Ethereum per validator), demonstrating their commitment to the network. This stake acts as a security deposit; validators who act against the network's rules or try to approve fraudulent transactions can lose part of their stake as a penalty, called slashing. This incentivizes good behaviour and secures the network.

There are more actors in PoS comparing to PoW process, this is by design to increase the decentralization and resiliency of the network. In the PoS process, a *block proposer* is comparable to a miner in PoW. The network algorithm selects a validator to propose a new block based on their stake and other factors like random selection for fairness [275]. The proposer's role is to gather transactions from the transaction pool (mempool), form them into a block, and propose it to the network. This block includes transaction data and a cryptographic hash of the previous block, maintaining the blockchain's integrity. The validator will receive a reward for proposing a block, which is typically a portion of the transaction fees included in the block and newly minted coins (ETH in the case of Ethereum).

The *block builder* is a role often seen in more advanced PoS systems. They are responsible for constructing the block—organizing and prioritizing transactions. In some systems, this role is separated from the block proposer to introduce another layer of decentralization and efficiency. By decoupling block construction from proposal, it allows for more specialized handling of transaction inclusion, potentially leading to optimized transaction throughput and fee market dynamics. Block builders are currently used as part of the MEV mechanism, which is explained more in 4.6.4. More over, a proposal to separate the block proposer and builder in Ethereum PoS is under consideration [341].

Relayers are participants in the network who help in relaying information between different parties or layers. In the context of PoS, they might be involved in transmitting block proposals, transactions, or validation results. Relayers play a crucial role in ensuring the smooth flow of information across the network, which is vital for maintaining the blockchain's speed and reliability. Currently, relayers are mainly used between the block proposers and builders in MEV infrastructure.

In summary, in a PoS system, validators stake cryptocurrency to participate in the process of block creation and validation. The block proposer and builder work together to form and optimize new blocks, while relayers facilitate communication within the network. This system aims to offer a more energy-efficient and potentially more decentralized alternative to PoW.

2.2 Ethereum

Ethereum [347] is a prominent public blockchain that has attracted the largest developer headcount compared to other blockchains. It is an extension of its predecessor, Bitcoin [234], but with significant enhancements, notably the addition of verbose *smart contracts*. These smart contracts are applications residing on the blockchain that can immutably execute their verified code. Verified code in this case means that anyone can compile the original source code and verify that the code deployed on the blockchain is the same code they compiled, there are also tools to automatically add a verification check to the smartcontract view (*e.g.,* etherscan.io). Ethereum operates on a Turing-complete virtual machine, the `Ethereum Virtual Machine (EVM)`, allowing programs to live and be executed on the blockchain. This differs from Bitcoin's UTXO[2] model, which primarily supports value transfers and has a scripting language for extending transaction functionality to a limited extent. In contrast, Ethereum's Turing-complete language opens up limitless possibilities. All transactions and executions on Ethereum are verified by a decentralized network of nodes. The nodes are incentivized to verify transactions and execute smart contracts by receiving rewards in the form of Ether, the native cryptocurrency of Ethereum.

Ethereum blockchain started in 2015 using the similar consensus mechanism as Bitcoin, called Proof of Work (PoW), also known as mining. However, in 2020, Ethereum started the transition to Proof of Stake (PoS) consensus mechanism. The main difference between PoW and PoS is that in PoW, the nodes are incentivized to solve a computationally hard puzzle to be able to add a block to the blockchain. However, in PoS, the nodes are incentivized to stake their Ether to be able to add a block to the blockchain. The PoS mechanism is more energy efficient and more scalable than PoW. Ethereum fully switched to PoS in September 2022 in an event called the *Merge*, and reduced its energy consumption by 99.5% [126]. In this dissertation, we do not directly focus on the consensus mechanisms, however, we will discuss the implications of the consensus mechanism on the blockchain security in chapter 4.

[2] Unspend Transaction Output

2.2.1 Smart Contracts

Smart contracts, a fundamental component of blockchain technology, particularly in the Ethereum ecosystem, represent a paradigm shift in how contractual agreements are executed and enforced. These contracts are essentially codebases that reside on the blockchain, acting as autonomous agents that carry out predefined functions when a user invokes the contract and certain conditions are met.

At their core, Ethereum smart contracts are collections of code and data (state) residing at a specific address on the Ethereum blockchain. Developed primarily in Solidity, a high-level language specifically designed for Ethereum, these contracts encapsulate a set of rules and automatically enforce them through the code. Solidity, with its syntax similar to JavaScript and C++, enables developers to write applications that implement self-enforcing business logic encapsulated within smart contracts, thereby removing the need for intermediaries.

There are two types of Ethereum addresses, Externally Owned Accounts (EOA) are controlled by private keys and are used to send transactions on the Ethereum network. Smart contracts are accounts that are controlled by code and are used to store and execute code on the Ethereum network. Smart contracts are compiled into bytecode and deployed on the Ethereum blockchain. Once deployed, they become immutable – their code cannot be altered, ensuring the integrity of the contract. Execution of a smart contract is triggered by transactions. These transactions are sent by Externally Owned Accounts, containing the necessary information to invoke specific functions within the contract. Every operation in a smart contract requires a certain amount of *gas*, a unit that measures the computational effort required to execute operations [127]. Users initiating transactions must supply enough ether, Ethereum's native cryptocurrency, to cover the gas costs, which are determined by the complexity of the operations and the current network demand. This mechanism prevents inefficient or malicious contracts from wasting network resources. Smart contracts maintain an internal state stored on the blockchain. This state can include variables, balances, and other contract-specific data. When a contract's function is executed, it can alter its state (*e.g.,* updating balances, changing variables and records) and can also interact with other contracts, thereby enabling complex decentralized applications (Composability and DeFi).

Gas in the Ethereum network is a nuanced concept, intricately tied to the operations or *OPCODES* executed by a smart contract. Each OPCODE within a contract has an associated gas cost, which reflects the computational complexity of that operation [130]. The total gas required for a transaction is the sum of the gas costs of all OPCODES executed during the transaction. However, predicting the exact amount of gas a transaction will consume is challenging due to the dynamic nature of smart contract interactions and the fluctuating state of the transaction volume in Ethereum network. The gas cost for a particular operation can vary, influenced by network congestion and the data being processed at the time of the transaction execution [292].

The mechanism for determining transaction fees in Ethereum operates similarly to an auction system, known as *gas auctions*. Users specify a *gas price* they are willing to pay per unit of gas, and validators prioritize transactions with higher gas prices. This market-driven approach allows for dynamic adjustment of transaction fees based on current network demand and transaction priority. However, it also introduces volatility, making it difficult to estimate the exact cost in fiat currencies, like CAD, for executing transactions. Roughly, the cost of gas can fluctuate significantly, impacted by factors such as network activity levels and the overall demand for computational resources on the Ethereum blockchain. This pricing model's complexity is particularly relevant to the discussion of front-running (See Chapter 4), where participants may bid higher gas prices to prioritize their transactions, potentially influencing the order of transaction execution and the final state of the blockchain.

While Ethereum smart contracts offer a wide range of functionalities, they are not without limitations. Their capabilities extend from creating tokenized assets and managing digital identities to executing decentralized finance (DeFi) transactions and running complex DApps. Use cases like decentralized gambling platforms, voting systems, and automated payroll services exemplify their versatility. However, the "unstoppable" nature of these contracts also poses challenges, particularly in terms of security and scalability. The code's immutability means that any flaws or vulnerabilities in the contract cannot be easily rectified post-deployment, emphasizing the need for rigorous testing and auditing pre-deployment. Security in smart contracts is paramount, as vulnerabilities can lead to significant financial losses. Common issues include reentrancy attacks, where a malicious contract can repeatedly call a function in the original contract, and problems arising from visibility and access control, where private data or functions can be unintentionally exposed.

As noted earlier, everything on a blockchain is compromised of transactions and blocks. The order of the transactions in each block indicates the order of events and smart contract executions in the Ethereum blockchain. Given that miners, and recently entities named *block builders*, are in control of the order, it is possible for these entities to reorder the transaction in a block, or even not include a transaction in a block for higher financial gain from the new order. This is the basics of blockchain front-running that we discuss in the next chapters.

2.2.2 P2P Network

The Ethereum network, envisioned as a global, decentralized computer, operates on a peer-to-peer (P2P) basis, a key feature that distinguishes it from traditional centralized systems. This P2P architecture is foundational to Ethereum's functionality, enabling a trustless and permissionless environment where anyone can participate in the network. In the core of the design principles of Ethereum's P2P network, there are three main concepts: decentralization, trustless interactions, and permissionless nature. These concepts are the foundation of the Ethereum network and are the main reasons for the success of Ethereum. More technical details on how the P2P network impacts the information flow is discussed in 5.5.1.

Decentralization and Accessibility: Ethereum's network is designed to be fully decentralized, eliminating any central point of control or failure. This decentralization ensures that the network remains resilient against attacks and censorship. It is worthy to note that these are the ideal properties and the current state of the network does not have full resiliency against these attacks. The accessibility of the network allows anyone to run a full node, contributing to the network's health and security.

Trustless Interactions: In Ethereum's P2P network, nodes interact without needing to trust each other. Trustlessness is achieved through cryptographic verification methods and consensus algorithms, ensuring that all transactions and blocks adhere to the network's rules without requiring mutual trust amongst participants.

Permissionless Nature: The network's permissionless design means that anyone can join and leave the network at will, participate in mining activities, and contribute to the network's consensus process. This open-access principle is fundamental to Ethereum's ethos of inclusivity and decentralization.

As Ethereum has grown in popularity, the network has faced scalability issues, leading to congestion and high transaction fees. Many layer-2 scaling solutions like rollups, are being developed to address these challenges.

2.2.3 Transactions

Transactions are the means by which users interact with the blockchain, initiating value transfers and executing smart contracts' functionalities. Transactions are cryptographically signed messages sent by externally owned accounts (EOA) to the network [128], containing the necessary information to transfer value to the recipient or to invoke specific functions within the contract. Each transaction includes the sender's signature, recipient's address, the amount of Ether to be transferred, the gas limit and price, the data associated with the transaction (*e.g.,* the function to be called and the input arguments), and a nonce. The gas price is the amount of Ether the sender is willing to pay per unit of gas (up to the gas limit), which is determined by the complexity of the transaction and the current network demand [127]. Transactions are broadcast to the network, propagated by all the nodes in the P2P network. These transactions will live in the nodes' mempool before being picked up by block builders (miners), validated and amended as part of a block to the blockchain. It should be mentioned that the finality of the included transactions depend on the technical details of the consensus mechanism [10, 254] and is outside the scope of this dissertation.

2.2.4 Nodes

The Ethereum network comprises various types of nodes, including archival nodes, full nodes, and light nodes. Archival nodes store everything including all the state transitions which are not required to verify the latest state of the blockchain. Full nodes store the entire blockchain, validate transactions and blocks, and enforce consensus rules. Light nodes, designed for less resource-intensive operations, download only the header chain and request necessary data from full nodes. Nodes communicate using a P2P protocol, exchanging information such as transactions, blocks, and node data. This communication is facilitated by protocols like the Ethereum Wire

Protocol [104], which manages the synchronization of node data and the propagation of new blocks and transactions.

2.2.4.1 Mempool

When an Ethereum transaction is initiated, it first undergoes validation by a participating node in the network. This validation process includes verifying the transaction's signature and ensuring the sender has sufficient funds. Post-validation, the node disseminates the transaction across the network. Prior to its inclusion in a blockchain block, this transaction resides in what is known as the "mempool" (or memory pool) of each node. The mempool functions as a holding area for all pending transactions. Due to the asynchronous nature of transaction reception, each node may have a differently ordered mempool, as transactions are received and stored in the order they arrive.

The mempool in Ethereum nodes plays a crucial role in the transaction processing mechanism. It acts as a sort of waiting room for transactions before they are confirmed and added to a block. Each node in the Ethereum network maintains its own mempool, and there is no universal consensus on the order of transactions within these individual mempools. This is primarily because transactions reach different nodes at different times, leading to varying sequences.

Block builders (previously miners), who are responsible for creating new blocks, typically select transactions from their mempool. They often prioritize transactions with higher gas fees, as this maximizes their profit from block rewards and transaction fees. The decentralized and varied nature of mempools allows block builders to reorder transactions. This can be exploited for financial gain, a practice known as "transaction reordering" or "front running." They may choose transactions based on the transaction fees offered, leading to scenarios where transactions with higher fees are processed faster, while others with lower fees might experience delays. The mempool is a dynamic component of the Ethereum network, constantly changing with the arrival of new transactions and the creation of new blocks. It reflects the fluid nature of the network's transactional throughput and is a critical element in understanding Ethereum's operational mechanics. As the network evolves, managing the mempool efficiently remains a key challenge, particularly in addressing issues like network congestion, transaction prioritization, and the ethical implications of transaction reordering.

Chapter 3

Cryptojacking: from Replacing Ads to Invisible Abuse

This chapter is based on the paper "A first look at browser-based Cryptojacking" [121] published in the 2018 IEEE Security & Privacy on the Blockchain (IEEE S&B), co-located with the IEEE European Symposium on Security and Privacy (EuroS&P) at University College London (UCL). This paper was supervised by Jeremy Clark, and co-authored by Andreas Leoutsarakos and Troy Mursch.

3.1 Introduction

Online advertisement business is a multi-billion dollar industry. It is the primary source of revenue for many websites and services. However, the current online advertisement model is broken. It is plagued with problems such as click-fraud, privacy violations, and malvertising. As a result, many websites have started to look for alternative sources of revenue. One such alternative is browser-based cryptocurrency mining, also known as cryptojacking. As we discuss this new model in this chapter, we also show how bad actors and hackers have started to abuse it by turning their unprivileged access to income streams for themselves. We discuss ethical implications of this technology, and classify cryptojacking that does not gain user consent as *invisible abuse*: "the intentional use of the invisible operations of a computer to engage in unethical conduct" [227], which is discuss further in Section 3.7.

3.2 Background and Motivation

Bitcoin [234] emerged more than a decade ago as an open source project, which mushroomed into a cryptocurrency sector collectively capitalized at over $500 billion USD[1]. Every day, people new to the concept of cryptocurrencies look for a quick and simple way to acquire some crypto-wealth. In the early days of Bitcoin, users on their personal computers could effortlessly acquire the currency through mining—a process Bitcoin uses to incentivize nodes to verify transactions as they are recorded in the blockchain. However, a second wave of mining technology saw users augmenting the CPU power of their computers with GPUs. Other groups of people deployed snippets of JavaScript code on websites that recruited their visitor's CPU power, often unknowingly, to mine for them as part of a bigger mining network (*i.e.,* a

[1]Coinmarketcap - Global Charts - Accessed: 2017-12-14 `https://coinmarketcap.com/`

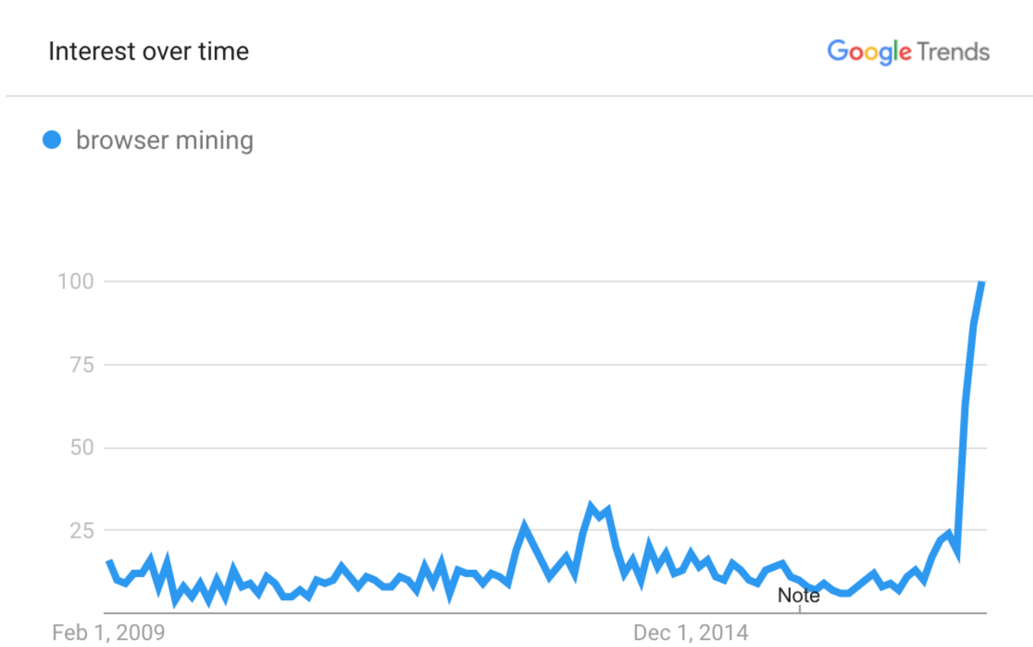

Figure 3.1: Search interest for "browser mining" over time. Search interest seems to have piqued during price surges, which culminated with Bitcoin crossing $1000 USD for the first time in December 2013. Soon after Bitcoin's first major crash searches consistently waned until a recent large spike, which is more than 4 times the lifetime average. The waning period before the recent surge could be attributed to the advent of ASIC usage for Bitcoin mining, and the surge is likely due to the revival of browser mining for non-Bitcoin currencies that have gained a sizeable market capitalization.

mining pool). However, both approaches quickly became infeasible as the computing power required to mine bitcoins grew exponentially to over 12 petahashes[2]. This was due to the emergence of application-specific integrated circuits (ASICs) and collective mining pools, which continue the third wave of mining to this day [236].

As the years passed and a few key cryptocurrencies emerged as the market leaders, the concept of browser mining largely became forgotten. Today, the most common way for the average person to acquire cryptocurrencies is to purchase them. It came as a surprise to many when stories began to circulate on popular media outlets this year

[2]Bitcoin hash rate - Accessed: 2017-11-20 `https://blockchain.info/charts/hash-rate`

about websites mining cryptocurrencies through browsers again. Figure 3.1 shows how the searches for "browser mining" have changed since Bitcoin was launched. Websites like The Pirate Bay [133] experimented with browser mining as a way to add a new revenue stream, while others like Showtime.com [318] claimed they had the code injected after they were discovered.

This chapter tells the story behind the rejuvenation of browser-based mining. It is centered on *cryptojacking* (also known as *coinjacking* and *drive-by mining*), a term coined to refer to the invisible use of a vulnerable user's computational resources to mine cryptocurrencies. Technically in-browser mining is a subset of cryptojacking, although most uses of the term apply to browser-based mining. In this case, mining happens within the client browser when the user visits the website. We have also seen the term cryptojacking applied to malware that mines cryptocurrencies, or in the situation where malware renders a machine as an unwitting participant in a botnet, and the botnet is rented for the purposes of mining cryptocurrencies (*cf.* [166]). The resource consumption of in-browser cryptojacking can noticeably degrade a computer's performance.

3.3 Preliminaries and Related Work

3.3.1 Browser-based Mining

3.3.1.1 Early days

The idea of in-browser mining started in the early days of Bitcoin. Bitcoin Plus[3] is one example of a discussion on replacing ads with Bitcoin browser miners[4]. It was also argued that browser-based mining provides greater scalability and decentralization as the barrier to entry is lowered to any unmodified computer with an internet connection. Soon after there was a rise in Bitcoin JavaScript miners such as JSMiner (2011)[5] and MineCrunch (2014)[6]. MineCrunch's visibility was increased by campaigns and the active online presence of its developers. Based on the developer claims, MineCrunch was well optimized for Javascript, but still worked 1.5x slower than native applications for CPU mining (*e.g.,* CPUMiner[7]). Although CPU mining became uncompetitive with GPU and ASIC-based mining, it remained a sandbox for botnet admins to experiment with the thousands of CPUs at their disposal. Botnet mining has been studied in the literature [166, 349], as well as covert mining within enterprises and cloud environments [303].

In addition to unprofitability, browser-based mining faced legal challenges. In May

[3]Bitcoin For the Uninitiated: Now, A Browser-Based Mining Client May 19th, 2011 https://www.themarysue.com/browser-based-bitcoin-mining/

[4]Bitcoin browser mining as a replacement for ads https://www.reddit.com/r/Bitcoin/comments/ieaew/bitcoin_browser_mining_as_a_replacement_for_ads/

[5]A JavaScript Bitcoin miner https://github.com/jwhitehorn/jsMiner

[6]MineCrunch, web(JS) miner with integration feature https://cryptocurrencytalk.com/topic/24618-minecrunch-web-js-miner-with-integration-feature/

[7]CPU miner for Litecoin and Bitcoin - https://github.com/pooler/cpuminer

2015, the New Jersey Attorney General's office reached a settlement with the developers of "Tidbit", a browser-based Bitcoin miner. Terms of the settlement included ceasing operations of Tidbit. Then acting Attorney General John J. Hoffman stated "No website should tap into a person's computer processing power without clearly notifying the person and giving them the chance to opt out." [244].

3.3.1.2 From One CPU to ASICS and Mining Pools

The first Bitcoin block mined on a GPU happened on July 18th, 2010 by a user named ArtForz [344], by using a private mining code that he developed himself. It was not until mid-2011 that others started implementing and releasing open source GPU-based mining tools. These tools greatly increased mining efficiency due to the hashing power of a GPU and the massive parallelizing possible with multiple GPUs (also known as mining rigs). The move from software to hardware followed shortly after. First, programmable FPGA chips resulting in custom-built circuits specifically for mining[8]. Then by mid-2012, companies started selling ASICs designed specifically for Bitcoin mining. After delay of about a year in delivering ASIC products, Bitcoin mining started transitioning from GPUs to ASICs where it remains today. Consequently, the hashing power of the Bitcoin network increased and the mining difficulty followed. To illustrate the change, consider a desktop PC CPU mining at 10 MH/s: on expectation, it will take 425 years before mining a single block [166].

In parallel to the evolving technology, collective action emerged through the use of

[8] Custom FPGA Board for Sale! (August 18, 2011) https://bitcointalk.org/index.php?topic=37904.0

mining pools. A mining pool is a collective of individual miners. Participants receive a slice of work for mining the current block on behalf of the pool. If a member of the pool mines the block, the block reward is split amongst the participants of the pool *pro rata* according to their computational effort [278]. As an aside, a very elegant protocol for reporting 'near-solutions' to the pool enables participants to prove, without trust, the level of effort they are contributing to the pool at all times. In general, a mining pools cannot amplify earnings, they only change their shape. An income stream from a pool is a steady trickle, while solo-mining results in sporadic dumps of income. The first Bitcoin block found on a mining pool was on December 16, 2010 that was a beta implementation of a pool operated by a user named *slush*.

3.3.2 Monero

Launched in April 2014, Monero [226] is a cryptocurrency alternative to Bitcoin. It purportedly offers increased privacy by obfuscating the participants in a transaction, as well as the amounts. This is in contrast to more popular cryptocurrencies like Bitcoin and Ethereum, where a pseudonymous-but-complete transaction graph can be constructed from the public blockchain. Some research has shown Monero's obfuscation techniques are less effective than originally claimed [221, 190]. Since regulation on exchanging between cryptocurrencies is lighter than exchanging cryptocurrencies for fiat money, and such services are not geographically bound, obtaining Monero for Bitcoin and vice versa is efficient and enables Monero to be used as a short-term medium of exchange for Bitcoin holders. This approach (and Monero's acceptance) is particularly popular on so-called dark web markets; markets that do not ban illicit goods and services.

A second characteristic that distinguishes Monero from Bitcoin is in the mining algorithm it uses. Monero still employs proof-of-work, specifically an algorithm called CryptoNight [92]. However, the computational puzzle is designed to be *memory-hard*: it requires the storage of a large set of bytes and then requires frequent reads and writes from this memory. Such puzzles are optimized for CPUs with low-latency memory-on-chip, and not as well suited for circuits like FPGAs and ASICs. CryptoNight requires approximately 2 MB per instance, which fits in the L3 cache of modern processors. Over the course of the next few years, these L3 cache sizes should become mainstream and allow more CPUs, and thus users, to participate in Monero's ecosystem. It has also been shown that ASICs cannot handle more than 1 MB of internal memory, which is less than the size of memory required to calculate a new block. GPUs are also at a disadvantage since GDDR5 memory, which are used in modern GPUs and considered one of the fastest types of memory, is notably slower than L3 cache [326].

3.3.3 Coinhive

Monero built on its early success and continued to gain in popularity over the years, which caught the attention of some developers who decided to revisit the idea of browser mining. See Table 3.1 for a timeline of events. One of the earliest efforts appeared in September 2017 and was called Coinhive [76]. Soon after, a competitor

2014-04-18	Monero Cryptocurrency released.
2017-09-14	Coinhive Miner launched.
2017-09-17	ThePirateBay caught using coinhive [311].
2017-09-21	Adblockers started to block coinhive scripts.
2017-09-24	Showtime caught running coinhive [318].
2017-09-25	Coinhive clones started to appear.
2017-10-13	PolitiFact website compromised [339].
2017-10-16	Coinhive launched authedmine - authorized mining.
2017-11-23	LiveHelpNow Hack incident [201].
2018-01-25	Cryptojacking code found on Youtube ads [219].
2018-02-11	UK Information Commissioner's Office incident [316].

Table 3.1: Timeline of Monero and In-browser Mining Reports

named Crypto-Loot[9] emerged. Both websites provided APIs[10] to developers for implementing browser mining on their websites that used their visitors' CPU resources to mine Monero. A portion of mined Monero would go back to the API developer, and the rest would be kept by the website. Not long after their early success, sev-

[9] Crypto-Loot - A web Browser Miner — Traffic Miner — CoinHive Alternative `https://crypto-loot.com/`

[10] Application Programming Interface

eral copycats appeared such as Coin-Have and PPoi [102] to take part in the reborn practice. It even inspired a new coin specifically designed for browser mining named JSECoin,[11] which has yet to find an audience. These developments took place over the course of a few weeks, which signalled the renewed success of browser mining. However, Coinhive's approach as a legitimate group set it apart from its peers and established itself as the leader in the space. They also launched separate services such as proof-of-work CAPTCHAs and short-links, which could be used to prevent spam while mining Monero [76].

3.4 Threat Model

In-browser mining is considered as an abuse unless user's consent is granted. The attack surface to abuse users' browsers through cryptojacking is broad, and there are multiple vectors where various entities can inject mining scripts in the website's codebase. We summarize those here.

3.4.1 Webmaster Initiated

A website administrator can add a mining script to her webpage, with or without informing users. Website owners may do this to monetize their sites, especially when they have been blacklisted or blocked by standard advertising platforms. In one example, a researcher found Coinhive on a large Russian website offering child pornography to users [282]. Revenue estimates, based on the website traffic data available, were

[11] JSEcoin's Website Cryptocurrency Mining `https://jsecoin.com/`

roughly \$10,000 a month after converting the value of XMR mined to USD.

3.4.2 Third-party Services

Many websites serve active JavaScript from third parties within their own webpages. This could be ads from an ad network, accessibility tools or tracking and analytics services. Third parties with these privileges can inject cryptojacking scripts into the sites that use them, either intentionally or as a result of a breach. The first two incidents, Coinhive was injected into the websites of Movistar[12] and Globovision[13] using Google Tag Manager[14]. Movistar stated that Coinhive was not put on their website by a hacker, but instead was due to *"internal error"* while they were conducting "production tests". No statement was provided by Globovision on why the cryptojacking scripts appeared on their site on November 15, 2017 [314]. Another high-profile cryptojacking case involving a Google platform occurred in January 2018 when Trend Micro researchers found advertisements containing Coinhive miner script were shown to YouTube users in Japan, France, Taiwan, Italy, and Spain for nearly a week [219]. Similar to Showtime, YouTube inherently has a high average visit duration as a video streaming site and thus is prime target for cryptojacking operations.

---

[12] Movistar is a major telecommunications brand owned by Telefonica, operating in Spain and in many Hispanic American countries `https://www.movistar.com/`

[13] Globovision is a 24-hour television news network in Venezuela and Latin America `http://globovision.com/`

[14] Google Tag Manager is a tag management system created by Google to manage JavaScript and HTML tags used for tracking and analytics on websites

3.4.3 Browser Extensions

Cryptojacking was not limited to websites in 2017. The Chrome extension *Archive Poster* remained on the Chrome Web Store for days while silently cryptojacking an unknown portion of their 100,000+ users. After multiple user reports, followed by multiple news media outlets covering the issue, the extension was removed [142]. Similar cryptojacking extensions has been identified on less popular Mozilla Firefox add-ons as well.

3.4.4 Breaches

If an attacker is able to breach principle servers, websites, extensions, or the scripting services they use, they can inject cryptojacking scripts that will impact the site's users without the site's knowledge or consent. For example, a researcher found a malicious modification to webchat system LiveHelpNow's SDK; it resulted in unsolicited mining across nearly 1500 websites using their chat support service [201] such as retail store chains Crucial and Everlast websites. In another example, Coinhive was found on the political fact-checking website PolitiFact[15] A compromised JavaScript library was found to be injecting the cryptojacking code. The malicious code remained on the site for at least four hours before it was removed [339]. PolitiFact executive director stated, "Hackers were able to install their script on the fact-checking website after discovering a misconfigured cloud-computing server" [333].

Another recent example of such incident is a breach in a website plugin called

[15] PolitiFact: Fact-checking US politics `https://politifact.com/`

Browsealoud[16] led to injection of cryptojacking scripts in some United Kingdom governmental websites such as *Information Commissioner's Office, UK NHS services, Manchester City Council* and around 4200 other websites [316]. Within the same month, cryptojacking script was seen on *Tesla* and *LA Times* websites through poorly secured cloud configuration [235].

3.4.5 Man-in-the-middle

A user's web traffic is often routed through intermediaries that may have plaintext access to content. For example, internet service providers or free public wireless routers can inject cryptojacking scripts into non-HTTPS traffic. Advertisement code injection has been seen in practice before [317] and there have been assertions of similar injections of browser mining scripts at certain Starbucks free Wi-Fi hotspots in Argentina[17].

3.5 Measurements

3.5.1 Prevalence of Coinhive and Alternatives

Based on the fact that Coinhive is the dominant website offering in-browser mining (see Figure 3.4), we first focus on measuring the prevalence of Coinhive scripts deployed on internet sites. We use the censys.io BigQuery dataset [110] for the top

[16] An accessibility tool to read the content aloud in multiple languages `https://www.texthelp.com/en-gb/products/browsealoud/`

[17] `https://twitter.com/imnoah/status/936948776119537665`

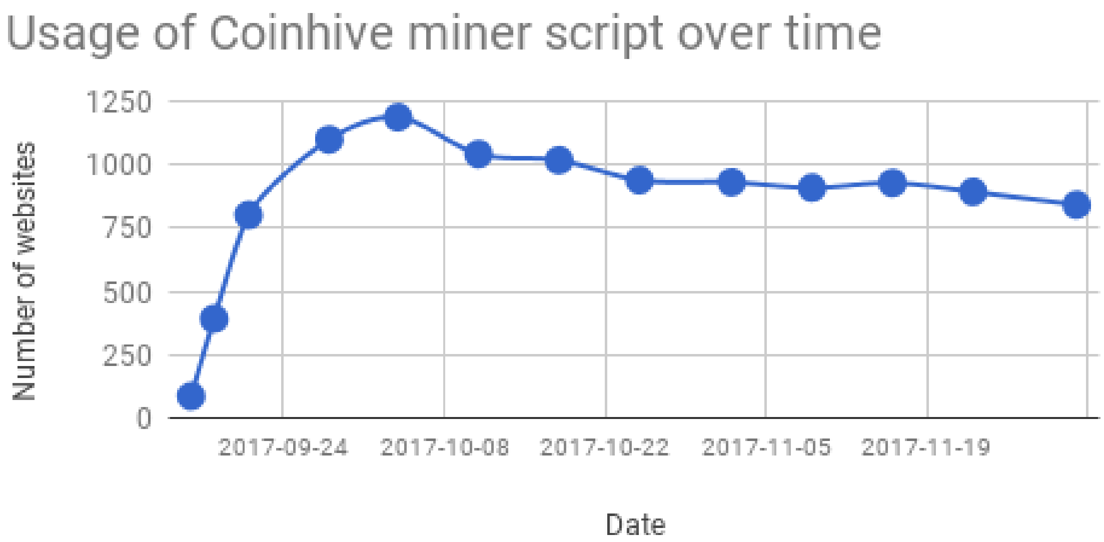

Figure 3.2: The number of instances of the Coinhive miner scripts found using the query in Figure 3.3 in top one million websites over a three month period beginning with the release of Coinhive in September 2017.

million sites indexed by Zmap[18]. We simply look for the `coinhive.min.js` script within the body of the website page. The query we use is in Figure 3.3 and the results over a two month period are provided in Figure 3.2. These findings are corroborated by another search engine, PublicWWW[19], which indexes the source code of publicly available websites. Using PublicWWW's dataset, over 30,000 websites were found to have the `coinhive.min.js` library [230]. As seen from our data in Figure 3.2, the adoption of this script was substantial in the first days of its release. However, progress slowed down at the same time as ad-blockers and organizations started to block Coinhive's website. The initial purpose of this service, as claimed by Coinhive, was to replace ads and cover server costs for webmasters. As the service

[18] `https://zmap.io`
[19] Search Engine for Source Code `https://publicwww.com/`

```sql
SELECT domain, tags, p80.http_www.get.headers.
    content_language, p80.http_www.get.headers.server, p80.
    http.get.headers.x_powered_by, p80.http.get.title, p80.
    http_www.get.body as wwwbody, p80.http.get.body as
    plainbody
FROM censys-io.domain_public.20171123
WHERE STRPOS(p80.http.get.body, coinhive.min.js) > 0 or
    STRPOS(p80.http_www.get.body, coinhive.min.js) >0)
```

Figure 3.3: A BigQuery SQL query to find websites that embed the Coinhive script using a dataset of the top one million sites from censys.io.

did not require that websites receive user consent before running the miner code, it started to be used maliciously in users‘ browsers. This type of usage resulted in Coinhive being included in some company's top-10 most wanted malware list [65].

This type of measurement will become less accurate moving forward. Cryptojacking services are evolving to use obfuscated JavaScript and randomized URLs to evade detection[20]. An example of these methods can be found in the cryptojacking service provider called Minr. In this case, the script is automatically obfuscated for users implementing the code. In addition, the domain names used by Minr frequently change to circumvent blocklists and anti-malware software.

Coinhive has begun to be blocked by enterprises. One example is shown in Figure 3.7. This blocking seems to have sent Coinhive operators to lesser known alternatives with the same or similar functionality. We used the same methodology on PublicWWW dataset to find the usage of Coinhive and its alternatives on the internet. Table 3.2 shows the keywords used to identify these services. The result can be found on Figure 3.4 and Figure 3.5.

[20] https://twitter.com/bad_packets/status/940333744035999744

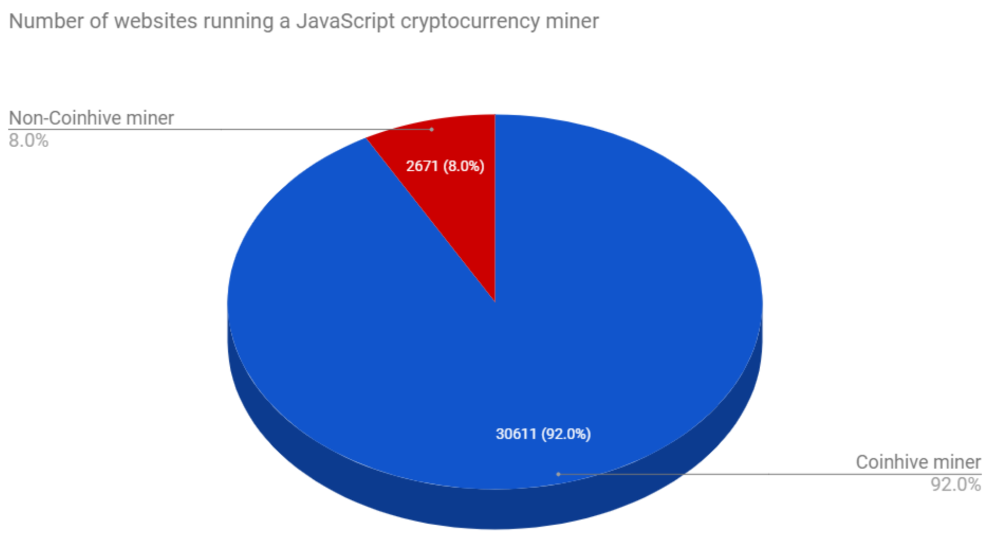

Figure 3.4: Share of websites using a JavaScript cryptocurrency miner, details in Table 3.2

Coinhive has also reacted by focusing on adding methods to enforce asking for user consent and legitimizing the use of cryptojacking. It introduced another domain and service called *Authedmine*, which requires user's consent to start mining in the browser. This service did not get the same attention as the original service, but it did inspire discussions regarding the ethics of such services, which is discussed in Section 3.7. Using the same methodology, censys.io was used to measure the prevalence of AuthedMine and show the results in Figure 3.8.

3.5.2 Client Impact

Most cryptojacking scripts discovered were configured to use around 25% of user's CPU, which can be justified as it will be under the threshold of attracting the user's

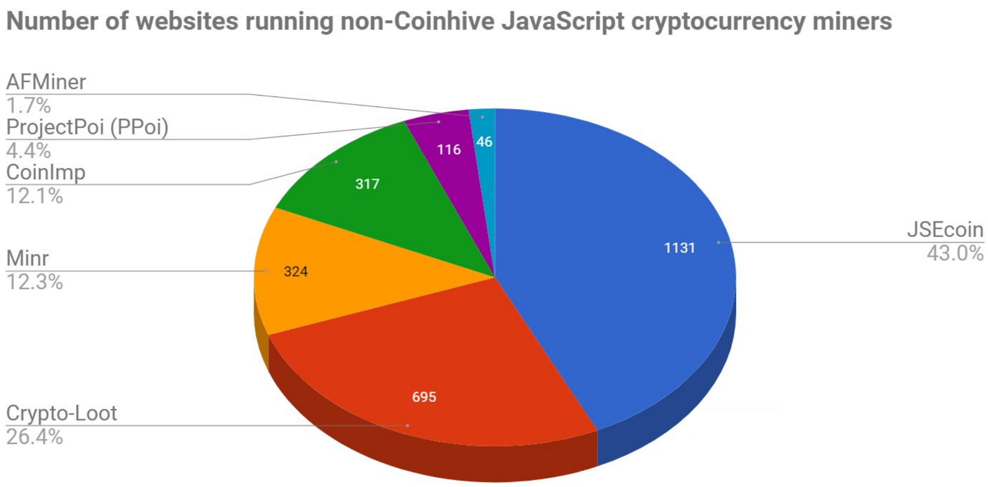

Figure 3.5: Share of websites using a Coinhive alternative, details in Table 3.2

attention, and it could be argued as fair-usage of their hardware. During the first few days, however, there were some reports of 100% CPU usage while visiting websites containing these scripts [312], which can be characterized as malicious. By default, the Coinhive JavaScript library will use all available CPU resources. The user implementing the script must include a throttle value to reduce the client-side CPU usage during mining operations. We show an example in Figure 3.9.

3.5.3 Profitability

Coinhive developers estimate a monthly revenue of about 0.3 XMR (about $101 USD) for a website with 10-20 active miners [76]. We sought to validate this estimation with a real world data set provided to us[21]. One of the biggest Coinhive campaign operators is a domain parking service. It runs Coinhive on over 11 000 parked websites.

[21] In collaboration and with thanks to Faraz Fallahi `https://github.com/fffaraz`

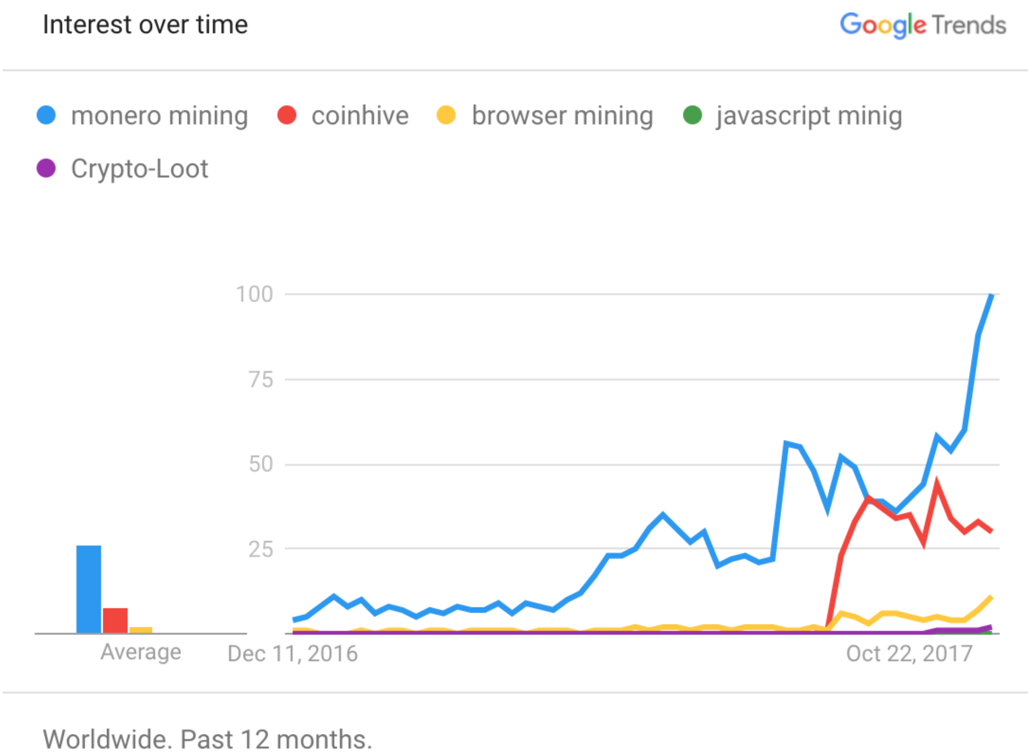

Figure 3.6: Google Trend over last 12 months: there has been more interest in Coinhive than the broader, related search term "Browser mining". Comparing to other services offering Monero browser mining API, Coinhive had the advantage of being the first to offer the service.

While visits to parked domains are considerably shorter than an average website, the data spans a period of three months and gives some insight into the profitability of cryptojacking. During the experimental period of about 3 months, they accumulated 105 580 user sessions for an average of 24 seconds per session. For the period examined, the revenue was 0.02417 XMR (Monero's currency) which at the time of writing is valued at $7.69 USD. Further detail is provide in Figures 3.10 and 3.11. While an A/B test was not setup to determine how much traditional web advertising would have brought in, freely available web calculator tools suggest we might expect an order or two of magnitude greater for comparable traffic.

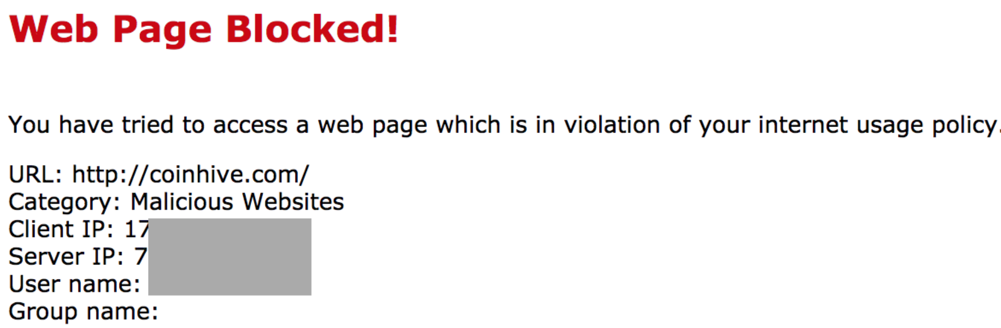

Figure 3.7: Concordia university has categorized the coinhive.com website as malicious and has blocked it.

3.6 Mitigations

We discuss the ethics of cryptojacking in the next section, but in the case of cryptojacking without user consent, it seems natural to us to presuppose users want to be protected. Protection might take a few forms, which we outline here.

3.6.1 Obtaining Consent

Cryptojacking tools might attempt to legitimize the practice by first obtaining user consent on a service provider level. An example of this is the Authedmine service from Coinhive discussed previously. Malicious sites might also opt for a service like Authedmine if it is whitelisted on its users‘ networks and then attempt to circumvent the consent process. For example, consent that requires a click from the user has been shown in some circumstances to be vulnerable to clickjacking attacks [283].

While cryptojacking is nowhere near the prevalence of tracking cookies, eventually it might grow into a regulatory issue where governmental bodies could use legislative approaches to obtain consent, similar to the provisions many countries now use for

Website	Results	Query Parameter
Coinhive	30611	‘coinhive.min.js‘
JSEcoin	1131	‘load.jsecoin.com‘
Crypto-Loot	695	‘CryptoLoot.Anonymous‘
Minr	324	‘minr.pw‘,‘st.kjli.fi‘, ‘abc.pema.cl‘,‘metrika.ron.si‘, ‘cdn.rove.cl‘,‘host.d-ns.ga‘, ‘static.hk.rs‘,‘hallaert.online‘, ‘cnt.statistic.date‘,‘cdn.static-cnt.bid‘
CoinImp	317	‘www.coinimp.com/scripts/min.js‘, ‘www.hashing.win‘
ProjectPoi (PPoi)	116	‘projectpoi.min‘
AFMiner	46	‘afminer.com/code/miner.php‘
Papoto	42	‘papoto.com/lib/papoto.js‘

Table 3.2: Cryptojacking data was gathered by totaling the number of websites which had the following libraries in their source codes, indexed by PublicWWW by 12/24/2017. Figure 3.4 and Figure 3.5 are visualizations of these results.

cookies (including honoring the ‘do not track’ HTTP header and obtaining click-based consent).

In addition to regulation and standardizing, better security design is required to narrow down the attack surface, such as using SSL/TLS and tokenizing the requests to make sure websites get user‘s consent before running these scripts. DNS TXT records could also be implemented to verify domain name ownership before browser mining scripts would be authorized to run on the website.

3.6.2 Browser-level Mitigation

Browser developers have begun discussion of intervening in cryptojacking[22]. Potential mitigation include: throttling client side scripting, warning users when client side

[22]‘Please consider intervention for high CPU usage js’ `https://bugs.chromium.org/p/chromium/issues/detail?id=766068`

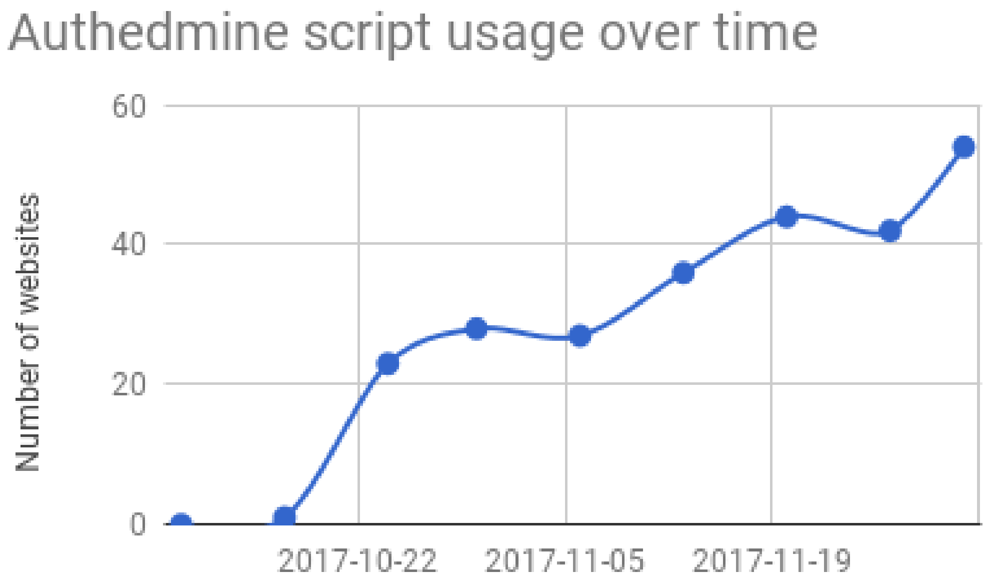

Figure 3.8: Usage of AuthedMine Miner scripts in top one million websites since its introduction

scripting consumes excessive resources, and blocking the sources of known cryptojacking scripts. Determining appropriate for thresholds for client-side processing that are high enough to allow legitimate applications and low enough to deter cryptojacking is an open research problem, as would be the wording of any notifications to the user that would lead the user to make an informed decision about allowing or not allowing resource consumption (*cf.* SSL/TLS warnings [299, 296, 2]). Browsers such as Opera, have taken a stance against cryptojacking scripts and blocked them via their "NoCoin" blacklist [248]. It is too early to determine the effectiveness of using a blacklist to block such activities. A few other detection mechanism that were released after our paper was published are discussed in 3.8.

It is worth noting that some browsers might actually take the exact opposite approach and promote (consensual) in-browser mining, as it enables a form of mon-

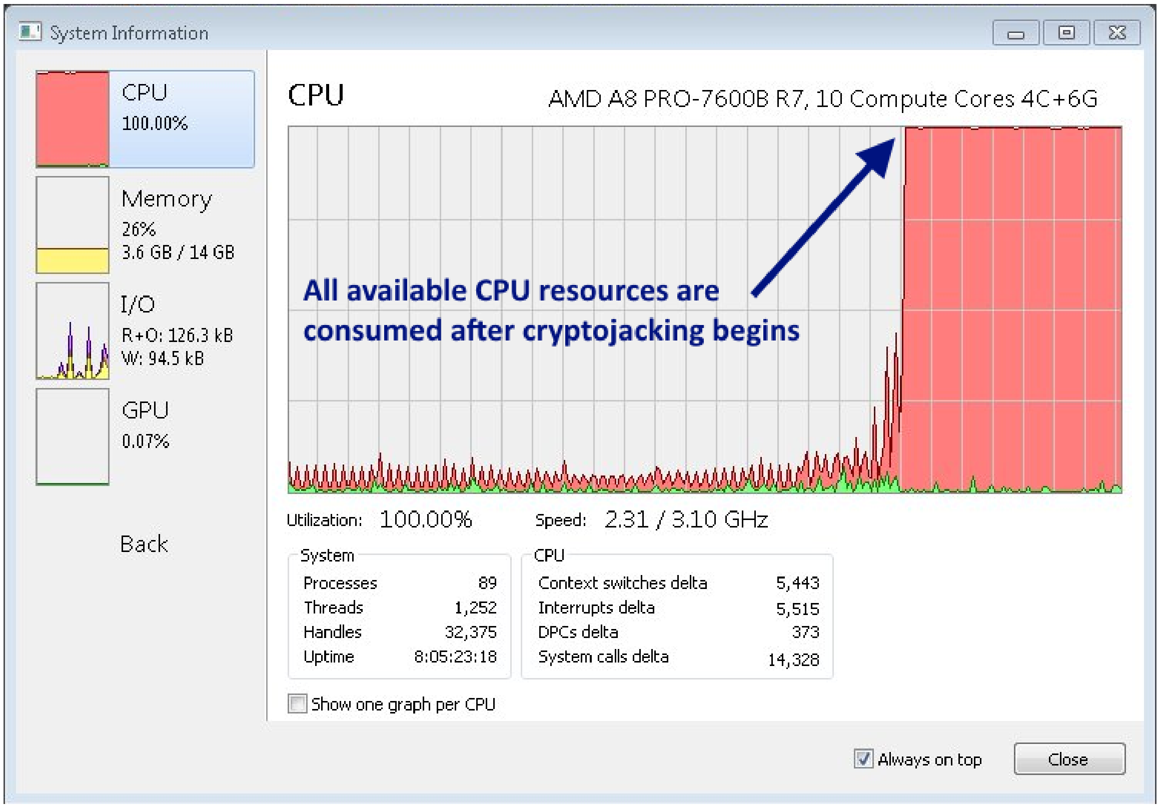

Figure 3.9: Comparison of CPU usage of browser without and with browser mining enabled.

etizing websites independent of both (1) ad networks and the user tracking that accompanies the current ad model, and (2) users maintaining some form of credits or currencies for making micropayment to websites they use(*e.g.*, Brave Browser [23]). Browser mining has been shown to not be as efficient as native mining applications today. Therefore, optimizations on how browsers pass system calls to the operating system can be made, or there can even be browsers designed specifically to support efficient browser mining.

[23] `https://brave.com`

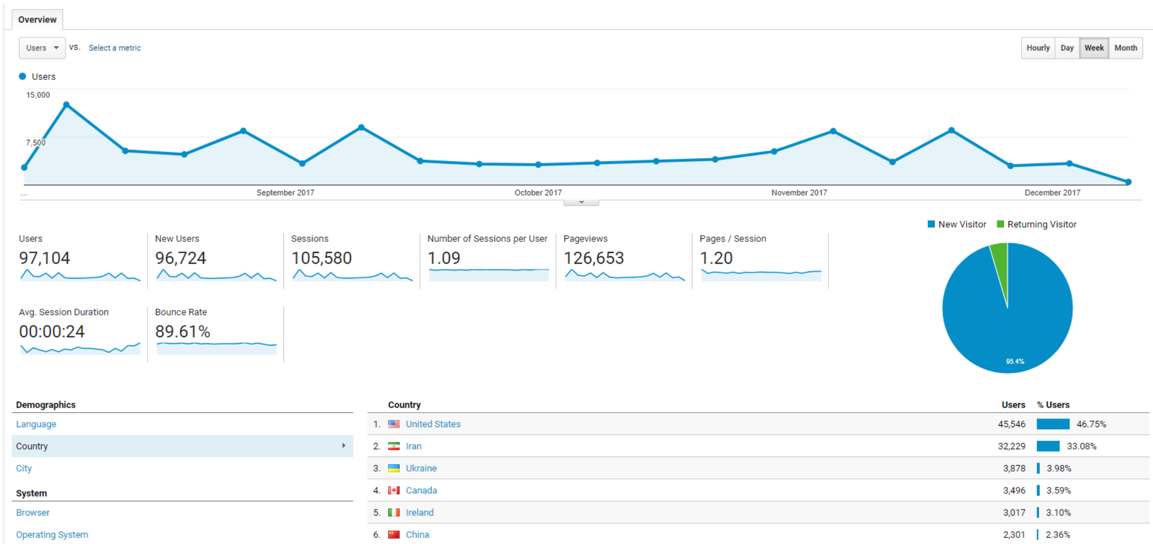

Figure 3.10: Google Analytics dashboard showing the number of visitors to a domain parking service of 11 000 domains.

3.7 Discussion

While cryptojacking might be relatively new, it fits the pattern of various other technologies deployed on the web that raise ethical questions. In thinking about it, we distinguish a few cases: (1) the use of cryptojacking on a breached website, (2) the use of cryptojacking by the website owner with an attempt at obtaining user consent, and (3) the use of cryptojacking by the website owner without obtaining user consent. We would argue that (1) is clearly unethical; invariant to one's views on the ethics of hacking, we cannot see a justification for a breach that profits the adversary without any external benefits to anyone else.

As for (2), cryptojacking after gaining user consent, is controversial primarily because it is unclear if users understand what they are consenting to, what they receive in return (some examples might include the elimination of ads, premium features,

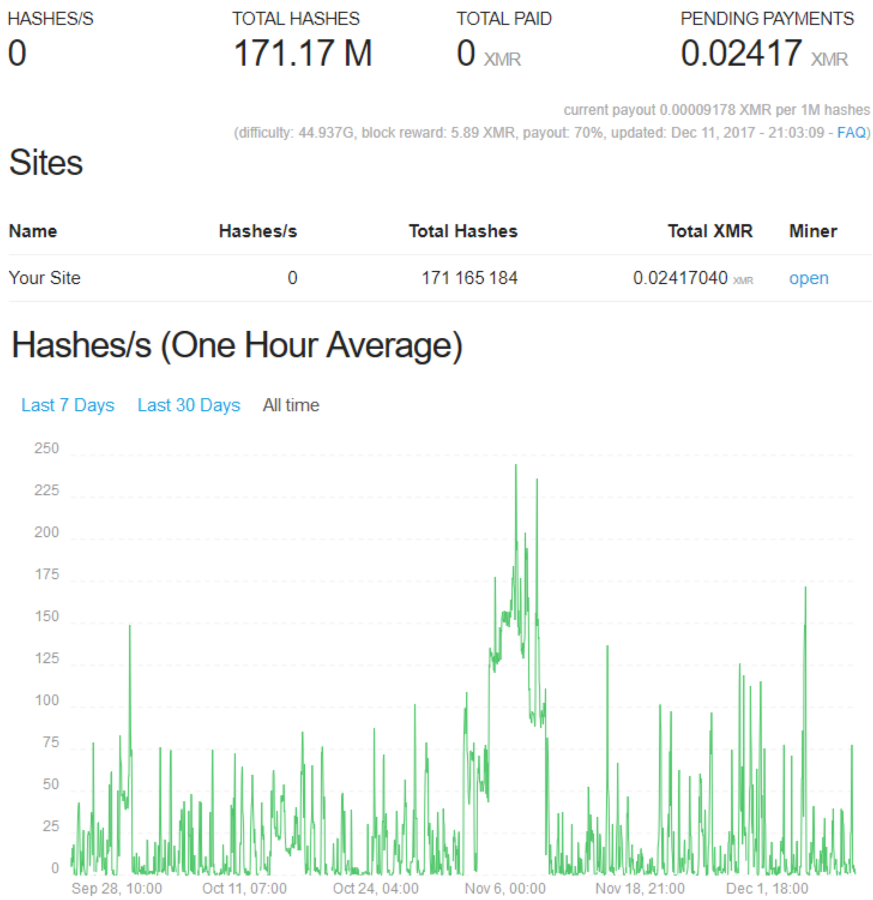

Figure 3.11: Coinhive dashboard showing the earnings of a domain parking service that runs Coinhive on 11 000 domains. Over the course of about 3 months, the operator earned 0.02417 XMR (currently $7.69 USD).

paywalled content, or higher definition video streams), and whether it is a fair exchange. To understand the zeitgeist, consider a recent poll conducted by Bleeping Computer that found: "many users said they are OK with websites mining Monero in the background if they don't see ads anymore" [36]. Coinhive released AuthedMine in recognition of the importance to many of user consent. ThePirateBay.org [26] ran cryptojacking scripts while users searched for torrent files without notice in their Privacy Policy, nor any visible warning on any part of the website that informed their users of this activity. This resulted in a backlash against the website, which responded with the following statement, "Do you want ads or do you want to give away a few of your CPU cycles every time you visit the site?" [312]. While the admins admitted to their testing of browser mining, their notice came after it was discovered and they ultimately removed the code. In both auction-based and keyword-based online advertisement, the advertiser pays the advertisement publisher to distribute the advertisement and the advertisement publisher pays a portion of the revenues to the website owner whom the advertisement was shown on her website [181]. However, with in-browser mining as a replacement monetization strategy, a more direct compensation is established with fewer intermediaries which could benefit users and sites alike.

The potential harm to users of cryptojacking is higher energy bills, along with accelerated device degradation, slower system performance, and a poor web experience [293, 311]. While consent may be obtained from the user, it is unclear if the user's mental model of how they are paying can be made clear to them. On the other hand, the privacy disclosures users make in the traditional advertising model are also intangible; it is doubtful users understand what they are consenting to when they, for example, consent through a banner [132] to the use of tracking cookies; and many websites waste computational resources without consequence of buggy scripting and unnecessary libraries. In short, the ethics are not clear-cut and should be debated.

One webservice prone to cryptojacking is video streaming—the longer a user is engaged on a website, the more income can be earned through browser mining. Showtime.com [313] and UFC.com [315] are two popular streaming sites that were asserted by researchers to have deployed Coinhive. Showtime has declined to comment on how or why Coinhive was implemented on their website. Speculation has been raised that it was injected via a third-party analytic tool, New Relic, due to Coinhive being found inside the New Relic code block within showtime's website source code. However, a New Relic representative denied these claims in a statement to The Register, "It appears [Coinhive scripts] were added to the website by [Showtime's] developers." [313]. In a statement released by the UFC, they denied the presence of the code stating, "[they] did not find any reference to the mentioned Coinhive JavaScript [code]"[24].

The third case –(3)– is the use of cryptojacking without user consent. Moor, in "What is Computer Ethics?" [227] introduces the concept of an *invisible factor* for invisible computer operations in society. Based on his definitions, we would classify cryptojacking that does not gain user consent as *invisible abuse*: the intentional use of the invisible operations of a computer to engage in unethical conduct. A few other examples of *invisible abuse*, as discussed in Moor's paper [227], is the case of a programmer stealing excess interest from a bank. When interest on a bank account is calculated, often a small fraction of a cent is left over, and the programmer can decide to transfer that fraction to his own account. Another example is regarding the use of computers for surveillance of employees, or modeling user behavior for targeted marketing. Here the cryptojacker is earning money from unaware users that are being charged on their electricity bill. As discussed before, we already have

[24] https://twitter.com/bad_packets/status/928044219222048769

court cases against such activities [244] and regulations for activities such as online user tracking [132], which indicates the need to start discussions and regulation on in-browser mining to fill in this policy vacuum as well.

3.8 Follow-up Work

Coinhive shutdown their official service on March 2019, and with that most of the cryptojacking scripts stopped working. However, due to the vast amount of copycats there are still some websites that are using cryptojacking scripts [327].

We were the first paper published on *cryptojacking*, and since then there has been over 200 papers citing our paper. We have seen a lot of research in this area, and we will discuss some main research avenues that have been explored since our paper was published.

Tekiner, Ege *et al.* published a systematization of knowledge (SoK) on all types of cryptojacking [308]. While some other papers covered a more detailed technical analysis and systematic study on the in-browser cryptojacking in the wild [164, 281, 32, 232, 231, 284].

Another avenue of research is on detection of cryptojacking presence using different methodologies, such as using heuristics and function detection [179, 184], CPU usage and behavioral analysis [148, 242, 277, 305], and machine learning [238, 58, 200, 258]. This avenue of research continues to grow with novel and more advanced detection methodologies.

Following up on our research, Papadopoulos *et al.* conducted a profitability analysis on a few cryptojacking campaigns and discussed user consent ethical issues [251]. They discuss the potential of ethical use of cryptojacking as a decent replacement for online advertisements. This question was initially the main reason we were interested in looking into Coinhive, which to this day is a relevant avenue for research [351].

3.9 Conclusion

In-browser mining was introduced as a potential replacement for online advertisements to remove the intermediaries and connect the user directly to the website. However, it was quickly abused by malicious actors and resulted in some unforeseen consequences. We still believe there's a potential for ethical use of in-browser mining as a replacement for online advertisements. However, it requires a lot of work to be done in this area, such as user education for proper consent, ethical framework for its use. It is clear that overall cryptocurrency regulations regarding such use cases are still in their infancy, if existent at all.

Chapter 4

Blockchain Front-running: from Transparency to Extracting Value

This chapter is based on the paper "SoK: Transparent Dishonesty: front-running attacks on Blockchain" [122] published at 3rd Workshop on Trusted Smart Contracts In Association with Financial Cryptography (FC) in February 2019. This paper was supervised by Jeremy Clark, and co-authored by Seyedehmahsa Moosavi.

4.1 Introduction

Front-running has been an issue in financial instrument markets since the 1970s. With the advent of the blockchain technology, front-running has resurfaced in new forms we explore here, instigated by blockchain's decentralized and transparent nature. Due to this technical nuance of blockchains, any actor can act as the actor with privileged information and abuse this early access to information to profit from it. In the financial markets, the privileged information is mostly regarding trades and financial data, however, as blockchains have other use cases, front-running attack are expanded to many other applications, as discussed in this chapter.

In this section, we briefly describe what front-running is in the traditional markets and how it is related to other concepts regarding its legality and ethics. Additionally, we discuss the background of front-running on the blockchain applications and how the advancement and complexity of the technology has increased the attack vector to applications that previously were not vulnerable to information flow attacks. Finally, we discuss the related literature in this area.

4.2 Background and Motivation

The mechanics of front-running work on all DApps but front-running is not necessarily beneficial, depending on the DApp's internal logic and/or as any mitigation it might implement. Therefore, DApps need to be studied individually or in categories. In this chapter, we draw from a scattered body of knowledge regarding front-running attacks on blockchain applications and the proposed solutions, with a series of case studies of DApps deployed on Ethereum (a popular blockchain supporting DApps). We do case studies on decentralized exchanges (*e.g.,* Bancor), crypto-collectibles (*e.g.,* CryptoKitties), gambling services (*e.g.,* Fomo3D), and decentralized name services (*e.g.,* Ethereum Name Service). We also study initial coin offerings (ICOs). Finally, we provide a categorization of techniques to eliminate or mitigate front-running including transaction sequencing, cryptographic techniques like commit/reveal, and redesigning the functioning of the DApp to provide the same utility while removing time dependencies.

4.2.1 Traditional Front-running

Front-running is a course of action where someone benefits from early access to market information about upcoming transactions and trades, typically because of a privileged position along the transmission of this information and is applicable to both financial and non-financial systems. Historically, floor traders might have overheard a broker's negotiation with her client over a large purchase, and literally race the broker to buy first, potentially profiting when the large purchase temporarily reduces the supply of the stock. Alternatively, a malicious broker might front-run their own client's orders by purchasing stock for themselves between receiving the instruction to purchase from the client and actually executing the purchase (similar techniques can be used for large sell orders). Front-running is illegal in jurisdictions with established securities regulation.

Cases of front-running are sometimes difficult to distinguish from related concepts like insider trading and arbitrage. In front-running, a person sees a concrete transaction that is set to execute and reacts to it before it actually gets executed. If the person instead has access to more general privileged information that might predict future transactions but is not reacting at the actual pending trades, we would classify this activity as insider trading. If the person reacts after the trade is executed, or information is made public, and profits from being the fastest to react, this is considered arbitrage and is legal and encouraged because it helps markets integrate new information into prices quickly.

4.2.2 Literature on Traditional Front-running

Front-running originates on the Chicago Board Options Exchange (*CBoE*) [211]. The Securities Exchange Commission *(SEC)* in 1977 defined it as: "The practice of effecting an options transaction based upon non-public information regarding an impending block transaction[1] in the underlying stock, in order to obtain a profit when the options market adjusts to the price at which the block trades. [288]" Self-regulating exchanges (*e.g., CBoE*) and the *SEC* spent the ensuing years planning how to detect and outlaw front-running practices [211]. The *SEC* stated: "It seems evident that such behaviour on the part of persons with knowledge of imminent transactions which will likely affect the price of the derivative security constitutes an unfair use of such knowledge.[2]" The *CBoE* tried to educate their members on existing rules, however, differences in opinion regarding the unfairness of front-running activities, insufficient exchange rules and lack of a precise definition in this area resulted in no action [288] until the SEC began the regulation. We refer the reader interested in further details

[1] A block in the stock market is a large number of shares, 10 000 or more, to sell which will heavily change the price.

[2] Securities Exchange Act Release No. 14156, November 19, 1977, (Letter from George A. Fitzsimmons, Secretary, Securities, and Exchange Commission to Joseph W. Sullivan, President CBoE).

on this early regulatory history to Markham [211]. The first front-running policies applied only to certain option markets. In 2002, the rule was expanded to cover all security futures [17]. In 2012, it was expanded further with the new amendment, FINRA Rule 5270, to cover trading in options, derivatives, or other financial instruments overlying a security with only a few exceptions [19, 18]. Similar issues have been seen with domain names [80, 111] as well.

4.2.3 Background on Blockchain Front-running

Blockchain technology (introduced via Bitcoin in 2008 [234]) strives to disintermediate central parties that participate in a transaction. However, blockchains also introduce new participants in the process of relaying and finalizing transactions. Miners are in the best position to conduct these attacks as they hold fine-grained control over the exact set of transactions that will execute and in what order and can mix in their own (late) transactions without broadcasting them. Miners do however have to commit to what their own transactions will be before beginning the proof of work required to solve a block.

Any user monitoring network transactions (*e.g.,* running a full node) can see unconfirmed transactions. On the Ethereum blockchain, users have to pay for the computations in a small amount of Ether called gas [129]. The price that users pay for transactions, gasPrice, can increase or decrease how quickly miners will execute them and include them within the blocks they mine. A profit-motivated miner who sees identical transactions with different transaction fees will prioritize the transaction

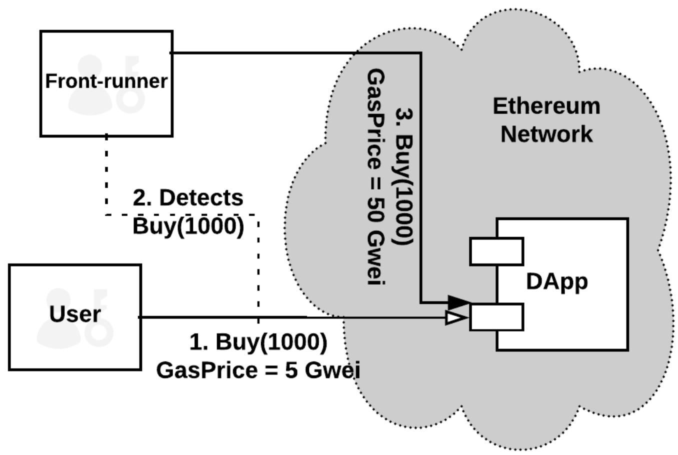

Figure 4.1: The front-runner upon spotting the profitable transaction *Buy(1000)* sends his own transaction with higher gas price to bribe the miners to prioritize his transaction over initial transaction.

that pays a higher gas price due to limited space in the blocks. This has been called a gas auction [167]. Therefore, any regular user who runs a full-node Ethereum client can front-run pending transactions by sending adaptive transactions with a higher gas price (see Figure 4.1).

Finally, well-positioned relaying nodes on the network (or part of the broader internet backbone) can attempt to influence how transactions are propagated through the network, which can influence the order miners receive transactions, or if they receive them at all [157, 210].

4.2.4 Literature on Blockchain Front-running

Front-running is related to two well-studied concepts: double-spending and rushing adversaries [185]. Double-spending attacks in Bitcoin are related to front-running [23, 176]. In this attack, a user broadcasts a transaction and is able to obtain some off-blockchain good or service before the transaction has actually been (fully) confirmed. The user can then broadcast a competing transaction that sends the same unspent coins to herself, perhaps using higher transaction fees, arrangements with miners or artifacts of the network topology to have the second transaction confirmed instead of the first. This can be considered a form of self-front-running. In the cryptographic literature, front-running attacks are modeled by allowing a so called 'rushing' adversary to interact with the protocol [28]. In particular, ideal functionalities of blockchains (such as those used in simulation-based proofs) need to capture this adversarial capability, assuming the real blockchain does not address front-running. See *e.g.,* Bitcoin backbone [146] and Hawk [185].

Aune *et al.* discuss how the lack of time priority between broadcasting a transaction and its validation by miners on a blockchain based system would lead to market information leakage [16]. They also propose a cryptographic approach, similar to commit and reveal (see Section 4.6.2) to prevent front-running.

Furthermore, Daian *et al.*, looks at the economically motivated front-running attacks by introducing the notion of *Maximum Extractable Value (MEV)* [98]. MEV[3] is defined by actions that facilitates transaction reordering or front-running attacks to profit from the transactions in the mempool. Additionally, they run economical analysis on the MEV occurrences in the wild.

[3]Originally, MEV was centered around Miner Extractable Value. However, with the subsequent development of block builders, the scope of MEV broadened to encompass all network participants.

4.3 A Taxonomy of Front-running Attacks

As we will illustrate with examples through-out this chapter, front-running attacks can often be reduced to one of a few basic templates. We emphasize what the adversary is trying to accomplish (without worrying about how) and we distinguish three cases: displacement, insertion, and suppression attacks. In all three cases, Alice is trying to invoke a function on a contract that is in a particular state, and Mallory will try to invoke her own function call on the same contract in the same state before Alice. See Figure 4.2 for a visual representation of the taxonomy.

In the first type of attack, a *displacement attack*, it is not important to the adversary for Alice's function call to run after Mallory runs her function. Alice's can be orphaned or run with no meaningful effect. Examples of displacement include: Alice trying to register a domain name and Mallory registering it first [174]; Alice trying to submit a bug to receive a bounty and Mallory stealing it and submitting it first [47]; and Alice trying to submit a bid in an auction and Mallory copying it.

In an *insertion attack*, after Mallory runs her function, the state of the contract is changed, and she needs Alice's original function to run on this modified state. For example, if Alice places a purchase order on a blockchain asset at a higher price than the best offer, Mallory will insert two transactions: she will purchase at the best offer price and then offer the same asset for sale at Alice's slightly higher purchase price. If Alice's transaction is then run after, Mallory will profit on the price difference without having to hold the asset.

Attack Type	Description	Example
Displacement	**Not important** to the adversary for original function call **to run after her function.**	Domain Name Registration
Insertion	**Important** to the adversary for original function call to **run after her function.**	Buy(_price), _price > Best offer
Suppression*	Run Function and **delay original function** call (Or any other)	Fomo3D

* A.K.A Block Stuffing

Variants	Description	Example
Asymmetric	**Different function than the original caller**	Cancellation Griefing
Bulk	**Run Large set of functions**	Capped ICO

Figure 4.2: Front-running Taxonomy

In a *suppression attack*, after Mallory runs her function, she tries to delay Alice from running her function. After the delay, she is indifferent to whether Alice's function runs or not. We only observe this attack pattern in one DApp and the details are quite specific to it, so we defer discussion until Section 4.4.3.

Each of these attacks have two variants, *asymmetric* and *bulk*. In some cases, Alice and Mallory are performing different operations. For example, Alice is trying to cancel an offer, and Mallory is trying to fulfill it first. We call this *asymmetric displacement*. In other cases, Mallory is trying to run a large set of functions: for example Alice and others are trying to buy a limited set of shares offered by a firm on a blockchain. We call this *bulk displacement*.

Furthermore, Torres *et al.* [321] expands on this taxonomy and conducts an analysis of the profitability of each type of front-running attacks in the wild. In their research they show that suppression attack has the highest reward but also the highest risk, while the displacement attack has no risk as it does not depend on the

DApp Category	**Names**	Rank
Exchanges	IDEX	1
	ForkDelta, EtherDelta	2
	Bancor	7
	The Token Store	13
	LocalEthereum	14
	Kyber	22
	0x Protocol	23
Crypto-Collectible Games (ERC-721 [113])	**CryptoKitties**	3
	Ethermon	4
	Cryptogirl	9
	Gods Unchained TCG	12
	Blockchain Cuties	15
	ETH.TOWN!	16
	0xUniverse	18
	MLBCrypto Baseball	19
	HyperDragons	25
Gambling	**Fomo3D**	5
	DailyDivs	6
	PoWH 3D	8
	FomoWar	10
	FairDapp	11
	Zethr	17
	dice2.win	20
	Ether Shrimp Farm	21
Name Services	**Ethereum Name Service**	24

Table 4.1: Top 25 DApps based on recent user activity from `DAppRadar.com` on September 4th, 2018. The DApps that are in bold are discussed in this paper.

victim's transaction. Insertion attack is the most popular type of attacks mainly on the decentralized exchanges.

4.4 Cases of Front-running in DApps

To find example DApps to study, we used the top 25 DApps based on recent user activity from `DAppradar.com` in September 2018.[4] User activity is admittedly an imperfect metric for finding the 'most significant' DApps: significant DApps might be lower volume overall or for extended periods of time (*e.g.,* ICOs, which we remedy by studying independently in Section 4.5). However, user activity is an objective criterion, data on it is available, and the list captures our intuition about which DApps are significant. It suffices for a first study in this area, and is preferable over an ad hoc approach. Using the dataset, we categorized the top 25 applications into 4 principal use cases. The details are given in Table 4.1.

4.4.1 Markets and Exchanges

The first category of DApp in Table 4.1 are financial exchanges for trading ether and Ethereum-based tokens. Exchanges such as EtherDelta[5], purport to implement a decentralized exchange, however, their order books are stored on a central server they control and shown to their users with a website interface. Central exchanges can front-run orders in the traditional sense, as well as re-order or block orders on their servers. 0xProtocol [338] uses *Relayers* which act as the order book holders and could front-run the orders they relay.

As seen in traditional financial markets, one method to manipulate the spot price

[4] List of decentralized applications `https://DAppradar.com/DApps`

[5] Also known as ForkDelta for the user interface: `https://forkdelta.app/`

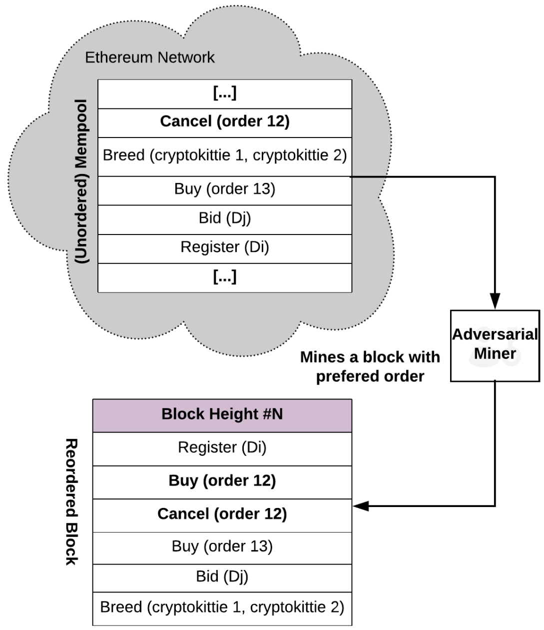

Figure 4.3: The adversarial miner monitors the Ethereum mempool for decentralized exchange transactions. Upon spotting a profitable cancellation transaction, he puts his buy order prior to the cancel transaction in the block he mines. Doing so, the miner can profit from the underlying trade and also get the gas included in the cancel transaction.

of an asset, is to flood the market with orders and cancel them when there are filling orders ("taker's griefing" [83]). Placing an order in a partially centralized exchange is free, but to prevent taker's griefing attacks, the user needs to send an Ethereum transaction to cancel each of his orders. Cancelling orders is most important when prices change faster than order execution. In this case, when an adversarial actor sees a pending cancellation transaction, he sends a fill order transaction with higher gasPrice to get in front of the cancellation order and take the order before it is canceled (this is known as *cancellation grief*). This attack follows the asymmetric displacement template and is illustrated in Figure 4.3.

Designing truly decentralized exchanges, where the order book is implemented directly on a public blockchain, is being pursued by a number of projects [107, 120]. These designs are generally vulnerable to front-running attacks following a displacement or insertion template. For example, a front-running full node or miner might gauge the demand for trades at a given price by the number of pending orders, and try to displace them at the same price assuming the demand is the result of the accurate new information about the asset. Alternatively, the front-runner might observe a large market order (*i.e.,* it will execute at any price). The adversary can try to insert a pair of limit orders that will bid near the best offer price and offer at a higher price. If the pair executes ahead of the market order, the front-runner profits by scalping the price of the shares. Finally, if adversary has pre-existing offers likely to be reached by the market order, she could insert cancellations and new offers at a higher price.

Front-running can also occur in non-financial smart contracts. As an example, a smart contract which adds all the participants to a party invite list could only be closed by the smart contract owner. In this case, when the owner sends the transaction to close the list, a miner can include his own list of participants before the close transaction in the block he mines. This is an issue with the design of the smart contracts and is known as *transaction-ordering dependence vulnerability* [205].

Bancor is an exchange DApp that allows users to exchange their tokens without any counter-party risk. The protocol aims to solve the cryptocurrency liquidity issue by introducing *Smart Tokens* [162]. Smart tokens are ERC20-compatible that can be bought or sold through a DApp-based dealer that is always available and implements a market scoring rule to manage its prices. Bancor provides continuous liquidity for digital assets without relying on brokers to match buyers with sellers. Implemented on the Ethereum blockchain, when transactions are broadcast to the network, they sit in a pending transaction pool known as *mempool* waiting for the miners to mine them. Since Bancor handles all the trades and exchanges on the chain (unlike other existing decentralized exchanges), these transactions are all visible to the public for some time before being included within a block. This leaves Bancor vulnerable to the blockchain race condition attack as attackers are given enough time to front-run other transactions, in which they can gain favorable profits by buying before the order and fill the original order with slightly higher price. Researchers have shown and implemented a proof of concept code to front-run Bancor as a non-miner user [38].

4.4.2 Crypto-Collectibles Games

The second category of DApp in Table 4.1 is crypto-collectables. Consider Cryptokitties [306], the most active DApp in this category and third most active overall. Each kitty is a cartoon kitten with a set of unique features to distinguish it from other cryptokitties, some features are rarer and harder to obtain. They can be bought, sold, or bred with other cryptokitties (Figure 4.4 shows an example of a Cryptokitty). At the Ethereum level, the kitty is a token implemented with *ERC-721: Non-Fungible Token Standard* [113]. Kitties are generally bought and sold on-chain through auction smart contracts. See Sections 4.4.1 and 4.4.4 for more details on auction-based front-running attacks.

ERC-721 are similar enough to ERC-20 tokens that they can be listed on exchanges

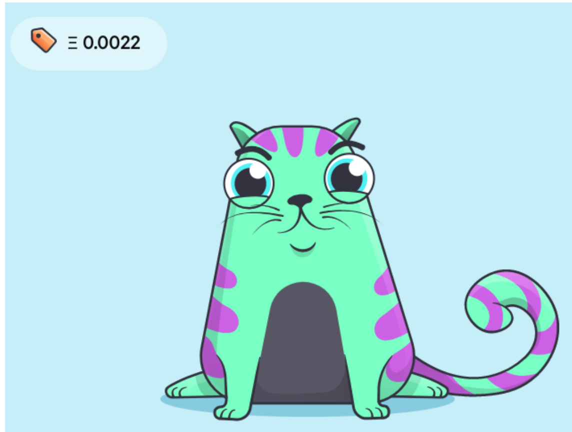

Figure 4.4: Cryptokitty # 842912 —Gen 8 —Snappy Cooldown

that are ERC-20 compatible. To buy a kitty from their main website, the user sends the following bid transaction: `bid(uint256 _tokenId)`. This is similar to open auctions and markets on section 4.4.1, bid value and the object (tokenId) bidding on is visible to the network and any user could easily front-run using insertion or displacement attack. The front-runner can see there is an interest in a specific kitty and insert a pair of transactions to buy it and re-auctioning it at a higher price, keeping any profit.

Specific to Cryptokitties protocol, they can breed and give birth. When cryptokitties breed, the smart contract sets from which future block the pregnancy of the cat can be completed. Anyone can complete the pregnancy by calling `giveBirth()` after the birthing block, and they will receive a reward in ether[6]. Even though front-running these calls would not affect the protocol workflow, but this displacement attack could result in financial profit for front-runners [269, 182].

[6]As there are no automated function calls in Ethereum, this incentive model –known as *Action Callback* [263]– is used to encourage users to call these functions.

4.4.3 Gambling

The third category of DApp in Table 4.1 is gambling services. While a large category of gambling games are based on random outcomes, DApps do not have unique access to an unpredictable data stream to harvest for randomness [260]. Any candidate source of randomness (such as block headers) is accessible to all DApp functions and can also be manipulated to an extent by miners.

Fomo3D is an example of a game style (known as Exit Scam[7]) not based on random outcomes, and it is the most active game on Ethereum in our sample. The aim of this game is to be the last person to have purchased a ticket when a timer goes to zero in a scenario where anyone can buy a ticket and each purchase increases the timer by 30 seconds. Many speculated such a game would never end but on August 22, 2018, the first round of the game ended with the winner collecting 10,469 Ether[8] equivalent to $2.1M USD at the time. Blockchain forensics indicate a sophisticated winning strategy to displace any new ticket purchases [12] that would reset the counter. The winner appears to have started by deploying many high gas consumption DApps unrelated to the game. When the timer of the game reached about 3 minutes, the winner bought 1 ticket and then sent multiple high gasPrice transactions to her own DApps. These transactions congested the network and bribed miners to prioritize them ahead of any new ticket purchases in Fomo3D. Recall this basic form of bribery is called a *Gas Auction*; See related work [215, 42] for more sophisticated bribery

[7] `https://exitscam.me/play`

[8] The first winner of Fomo3D, won 10,469 Ether `https://etherscan.io/tx/0xe08a519c03cb0aed0e04b33104112d65fa1d3a48cd3aeab65f047b2abce9d508`

contracts.

We classify this in the unique category of a suppression attack in our taxonomy (see Section 4.3). At first glance, it seemed like an extreme version of an asymmetric/bulk displacement attack on any new ticket purchase transactions. However, the key difference is that the front-runner does not care at all about the execution of her transactions—if miners mined empty blocks for three minutes, that would also be acceptable. Thus, bulk displacement[9] is simply a means-to-an-end and not the actual end goal of the adversary.

4.4.4 Name Services

The final category in Table 4.1 is name services, which are primarily aimed at disintermediating central parties involved in web domain registration (*e.g.,* ICAAN and registrars) and resolution (*e.g.,* DNS). For simple name services (such as some academic work like Ghazal [228]), domains purchases are transactions and front-runners can displace other users attempting to register domains. This parallels front-running attacks seen in regular (non-blockchain) domain registration [80]. Ethereum Name Service (ENS) [171] is the most active naming service on Ethereum. Instead of allowing new `.eth` domain names to be purchased directly, they are put up for a sealed bid auction which seals the domain name in a bid, but not the bid amount. Most implementations use the more user-friendly but less confidential method for starting and bidding on a domain name: `startAuctionsAndBid()`. This method leaks the hash of

[9]Also known as Block Stuffing Attack [294]

the domain and the initial bid amount in the auction. Original names can be guessed from the hashes (*e.g.,* rainbow tables, used in ENS Twitter bot[10]) or people can bid on domains even though they do not know what they are because of speculation on its value.

Users are allowed to bid for 3 days before the 2-day reveal phase begins (see 4.6.2), in which all bidders (winners and losers) must send a transaction to reveal their bids for a specific domain or sacrifice their bid amount. Also note that if two bidders bid the same price, the first to reveal wins it [106]. Using the leaked information, the domain squatter can win the auction with the same price of the original bidder by revealing it first. This is similar to front-running as it relies on inserting an action before the user, however we do not consider this specific action as front-running attack.

4.5 Cases of Front-running in ICOs

Initial coin offerings (ICOs) have changed how blockchain firms raise capital. More than 3000 ICOs have been held on Ethereum, and the market capitalization of these tokens appears to exceed $75B USD in the first half of 2018 [355]. At the DApp level, tokens are offered in short-term sales that see high transaction activity while the sale is ongoing and then the activity tapers off to occasional owner transfers. When we collected the top 25 most active DApps on `DAppRadar.com`, no significant ICOs were being sold. The ICO category slips through our sampling method, but we identify it

[10] `https://twitter.com/ensbot`

as a major category of DApp and study it here.

4.5.1 *Status.im* ICO

To deal with demand, ICOs cap sales in a variety of ways to mitigate front-running attacks. In June 2017, *Status.im* [307] started its ICO and reached the predefined cap within 16 hours, collecting close to 300,000 Ether. In order to prevent wealthy investors purchasing all the tokens and limit the amount of Ether deposited in each investment, they used a *fair* token distribution method called *Dynamic Ceiling* as an attempt to increase the opportunity for smaller investors. They implemented multiple caps (ceilings) in which, each had a maximum amount that could be deposited in. In this case, every deposit was checked by the smart contract and the exceeding amount was refunded to the sender while the accepted amount was sent to their multi-signature wallet address [259].

During the time frame the ICO was open for participation, there were reports of Ethereum network being unusable and transactions were not confirming. Further study showed that some mining pools might have been manipulating the network for their own profit. In addition, there were many transactions sent with a higher gas price to front-run other transactions, however, these transactions were failing due to the restriction in the ICO smart contract to reject transactions with higher than 50 *GWei* gas price (as a mitigation against front-running).

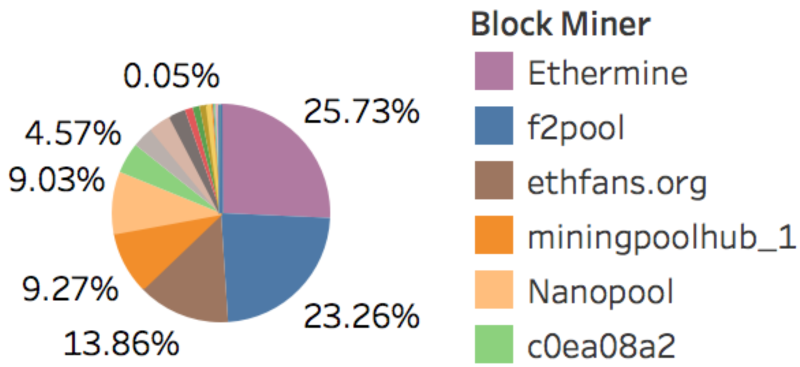

Figure 4.5: The percentage of Ethereum blocks mined between block 3903900 and 3908029, this is the time frame in which Status.im ICO was running. This percentage roughly shows the hashing power ratio each miner had at that time.

4.5.2 Data Collection and Analysis

According to the analysis we carried out, we discovered that the F2Pool—an Ethereum mining pool that had around 23% of the mining hash rate at the time (Figure 4.5)—sent 100 Ether to 30 new Ethereum addresses before the Status.im ICO started. When the ICO opened, F2Pool constructed 31 transactions to the ICO smart contract from their addresses, without broadcasting the transactions to the network[11]. They used their entire mining power to mine their own transactions and some other potentially failing high gas price transactions.

Ethereum's blockchain contains all transaction ever made on Ethereum. While the default client and online blockchain explorers offer some limited query capabilities, in order to analyze this case, we built our own database. Specifically, we used open source projects such as Go Ethereum implementation[12] for the full node,

[11] Note that we do not have an authoritative copy of the mempool over time, however, the probability of these transactions being broadcasted to the network and exclusively get mined by the same pool as the sender is low.

[12] Official Go implementation`https://github.com/ethereum/go-ethereum`.

a python script for extracting, transforming and loading Ethereum blocks, named `ethereum-etl` [217] and Google BigQuery.[13] Using this software stack, we were able to isolate transactions within the Status.im ICO. We used data analysis tool `Tableau`.[14] A copy of this dataset and the initial findings can be found in our GitHub repository[15].

As shown in Figure 4.6, most of the top miners in the mentioned time frame, have mined almost the same number of failed and successful transactions which were directed toward Status.im token sale, however F2Pool's transactions indicate their successful transactions were equivalent to 10% of the failed transactions, hence maximizing the mining rewards on gas, while censoring other transactions to the token sale smart contract. The terminology used here is specific to smart contract transactions on Ethereum, by *"failed transaction"* we mean the transactions in which the smart contract code rejected and threw an exception and by *"successful transaction"* we mean the transactions that went through and received tokens from the smart contract.

By tracing the transactions from these 30 addresses, we found explicit interference by F2Pool[18] in this scenario. As shown in Figure 4.7, the funds deposited by F2Pool in these addresses were sent to *Status.im* ICO and mined by F2Pool themselves, where the dynamic ceiling algorithm refunded a portion of the deposited funds. A few days

[13] `https://cloud.google.com/bigquery/`

[14] `https://www.tableau.com/`

[15] `http://bit.ly/madibaFrontrunning`

[17] Graph was made using `Blockseer.com` blockchain explorer.

[18] F2Pool address was identified by their mining reward deposit address `https://etherscan.io/address/0x61c808d82a3ac53231750dadc13c777b59310bd9`.

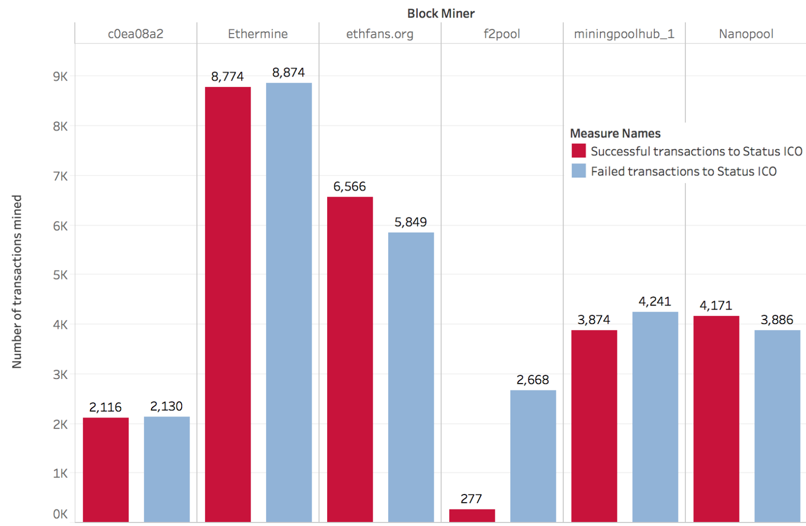

Figure 4.6: This chart shows the miner's behavior on the time frame that Status.im ICO was running. It is clear that the number of successful transactions mined by F2Pool do not follow the random homogeneous pattern of the rest of the network.

after these funds were sent back to F2Pool main address and the tokens were aggregated later in one single address. Although this incident does not involve transaction reordering in the blocks, it shows how miners can modify their mining software to behave in a certain way to front-run other transactions by *bulk displacement* to gain monetary profit.

4.6 Key Mitigations

As we studied front-running attacks on the blockchain, we also encountered a number of ways of preventing, detecting or mitigating front-running attacks. Instead of providing the details of exact solutions which will change over time, we extract the main principles or primitives that address the attack. A particular system may implement more than one in a layered mitigation approach.

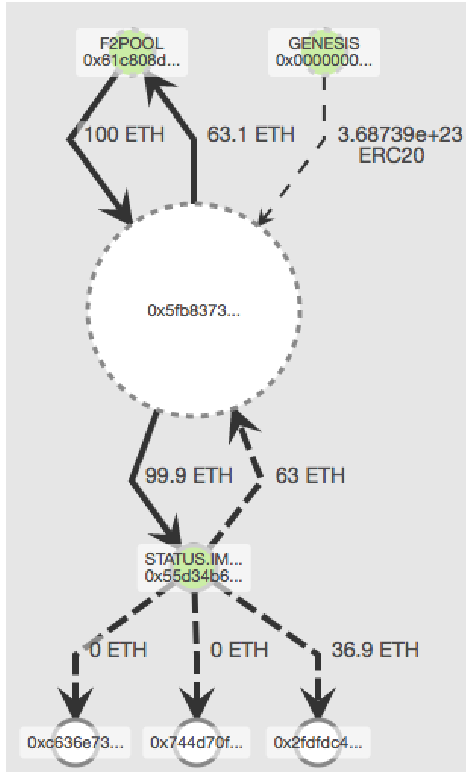

Figure 4.7: Prior to *Status.im* ICO *F2Pool* deposited 100 Ether in multiple new Ethereum addresses. On the time of the ICO, transactions sent from these addresses to *Status* ICO smart contract were prioritized in their mining pool, resulting in purchasing *ERC20* tokens. This method was used to overcome the dynamic ceiling algorithm of the ICO smart contract. Later on they sent the refunded Ether back to their own address.[17]

We classify the mitigations into four main categories. In the first category, the blockchain removes the miner's ability to arbitrarily order transactions and tries to enforce some ordering, or queue, for the transactions. In the second category, cryptographic techniques are used to limit the visibility of transactions, giving the potential front-running less information to base their strategy on. In the third category, DApps are designed from the bottom-up to remove the importance of transaction ordering or time in their operations. We also note that for DApps that are legally well-formed (*e.g.,* with identified parties and a clear jurisdiction), front-running attacks can violate laws, which is its own deterrent. The idea for the forth category is to embrace the front-running opportunities and design methods to share the profit with more actors than just the block builders. We discuss each of these categories in more detail below.

Traditional Front-running Prevention Methods. There are debates in traditional markets regarding the fact that front-running is considered to be a form of insider trading which deemed to be illegal. Traditional methods to prevent front-running mainly involves after the fact investigation and legal action against the front-runners [137]. As mentioned in section 4.2.2, defining front-running and educating the employees were the first step taken to prevent such issues in traditional markets, however, front-running became less likely to happen mainly because of the high fine and lawsuits against firms who behaved in an unethical way. Other methods such as dark pools [363, 56] and sealed bids [271] were discussed and implemented in a variety of regulated trading systems. The traditional methods to prevent front-running does not apply to blockchain applications, as mainly they are based on central enforcement and limitations, also in case of blockchains the actors who are front-running could be anonymous and the fear of lawsuits would not apply.

4.6.1 Transaction Sequencing

Ethereum miners store pending transactions in pools and draw from them when forming blocks. As the term 'pool' implies, there is no intrinsic order to how transactions are drawn and miners are free to sequence them arbitrarily.[19] The vanilla Go-Ethereum (geth) implementation prioritizes transactions based on their gas price and nonce [125]. Because no rule is enforced, miners can sequence transactions in advantageous ways. A number of proposals attempt to thwart this attack by enforcing a rule about how to sequence transactions.

First-in-first-out (FIFO) is generally not trivial to implement on a distributed network because transactions can reach different nodes in a different order. Kelker *et al.* proposes a solution this problem by adding a property to the consensus protocol – transaction-order-fairness– [178, 177]. Furthermore, A trusted third party can be used to assign sequential numbers to transactions (and sign them), but this is contrary to blockchain's core innovation of distributed trust. Nonetheless, most of the scaling layer two solutions use centralize sequencers and some exchanges do centralize time-sensitive functionalities (*e.g., EtherDelta* and *0xProject*) in off-chain order books [338, 337].

One alternative is to sequence transactions pseudorandomly. This can be seen in proposals like Canonical Transaction Ordering Rule (CTOR) by Bitcoin Cash ABC [329] which adds transactions in lexicographical order according to their hash [330]. Note that Bitcoin does not have a front-running problem for standard transactions. While this could be used by Ethereum to make front-running statistically difficult, the protection is marginal at best and might even exacerbate attacks. A front-runner can construct multiple equivalent transactions, with slightly different values, until

[19] Sometimes the pool is called a 'queue.' It is important to note is a misnomer as queues enforce a first-in-first-out sequence.

she finds a candidate that positions her transaction a desirable location in the resulting sequence. She broadcasts only this transaction and now miners that include her transaction will position it in front of transactions they heard about much earlier.

Finally, transactions themselves could enforce order. For example, they could specify the current state of the contract as the only state to execute on. This transaction chaining only prevents certain types of front-running; *i.e.,* it prevents insertion attacks but not displacement attacks (recall our taxonomy in Section 4.3). As transaction chaining only allows one state-changing transaction per state, at most one of a set of concurrent transactions can be confirmed; a drawback for active DApps.

Da Silva *et al.* [94] proposes an algorithm, known as Fixed Transaction Ordering and Admission (FTOA), in the consensus protocol that enforces the order of the transactions in the mined blocks, using logical timestamps in a Byzantine setting, to eliminate transaction reordering by miners. This is a theoretical solution and is not trivial to implement in the current Byzantine consensus protocols. Additionally, Kelkar *et al.* [178] proposes a solution by introducing an additional property to the Byzantine consensus protocol, transaction order-fairness, and they analyze the assumptions necessary to realize fair ordering. Another solution is to create a centralized sequencer that orders that transactions, this is a common solution for many Layer two scaling solutions on Ethereum. This solution is not ideal to decentralized applications as it requires a trusted third party to order the transactions. In some cases such as Fomo3D, the fair ordering of the transaction can increase the likelihood of the hacker winning the game by using the similar suppression attack as described in 4.4.3. One aspect of fair ordering proposals that is not discussed is the fact that some DApps might not be able to function as designed if the transactions are ordered "fairly". For example, Fomo3D game is designed to be unfair by including the protocol gas auctions as part of the game, and the fairness of the transactions would result in the game not functioning as intended.

4.6.2 Confidentiality

Privacy-Preserving Blockchains. All transaction details in Bitcoin are made public and participant identities are only lightly protected. A number of techniques increase confidentiality [52, 213] and anonymity [220, 243, 287] for cryptocurrencies. A current research direction is extending these protections to DApps [345, 276]. It is tempting to think that a confidential DApp would not permit front-running, as the front-runner would not know the details of the transaction she is front-running. However, there are some nuances here to explore.

A DApp interaction includes the following components: (1) the code of the DApp, (2) the current state of the DApp, (3) the name of the function being invoked, (4) the parameters supplied to the function, (5) the address of the contract the function is being invoked on, and (6) the identity of the sender. Confidentiality applied to a DApp could mean different levels of protection for each of these (See Figure 4.8). For front-running, function calls (3,4) are the most important, however, function calls could be inferred from state changes (2). Hawk [185] and Ekiden [67] are examples of (2,3,4)-confidentiality (with limitations we are glossing over).

1	**Code** of the DApp
2	Current **state** of the DApp
3	Name of the **function** being invoked
4	**Parameters** supplied to the function
5	**Address** of the contract the function is being invoked on
6	Identity of the **sender**.

Figure 4.8: Components Related to Any DApp Interaction

The applicability of privacy-preserving blockchains needs to be evaluated on a case-by-case basis. For example, one method used by traditional financial exchanges in dealing with front-running from high frequency traders is a dark pool: essentially a (2,3,4)-confidential order book maintained by a trusted party. A DApp could disintermediate this trusted party. Users whose balances are affected by changes in the contract's state would need to be able to learn this information. Further, if the contract addresses are known (*i.e.,* no 5-confidentiality), front-runners can know about the traffic pattern of calls to contracts which could be sufficient grounds for attack; for example, if each asset on an exchange has its own market contract, this leaks trade volume information. As a contrasting example, consider again decentralized domain registration: hiding state changes (2-confidentiality) defeats the entire purpose of the DApp, and protecting function calls is ineffective with a public state change since the state itself reveals the domain being registered.

Privacy-preserving blockchains strive (to some extent [222, 175]) to keep all details of transactions private, including participants and amounts. As an example, ZCash [165] uses two distinct types of addresses, transparent and shielded addresses.

Transparent addresses work similar to Bitcoin transactions, fully transparent about the sender and receiver addresses, the amount and included data. However, shielded addresses are private and do not leak any information. While this solution works for simple transactions, building similar shields for DApps is subject to ongoing research. With the possibility of having private smart contracts in such a setting, it could be feasible to achieve a front-run resistance blockchain, however, the functionality of the smart contracts could be limited.

Commit/Reveal. While confidentiality appears insufficient for solving domain name front-running alone, a hybrid approach of sequencing and confidentiality can be effective and is, in fact, an example of an older cryptographic trick known as commit/reveal. The essence of the approach is to protect the function call (*e.g.,* (3,4)- or (4)-confidentiality) until the function is enqueued in a sequence of functions to be executed. Once the sequence is established, the confidentiality is lifted and the function can only be executed in the order it was enqueued (or, generally speaking, not at all).

Recall that a commitment scheme enables one to commit to a digital value (*e.g.,* a statement, transaction, data, *etc.*) while keeping it a secret (*hiding*), and then open it (and only it: *binding*) at a later time of the committer's choosing [43]. A common approach (conjectured to be hiding) is to submit the cryptographic hash of the value with a random nonce (for low entropy data) to a smart contract, and later reveal the original value and nonce which can be verified by the contract to correctly hash to

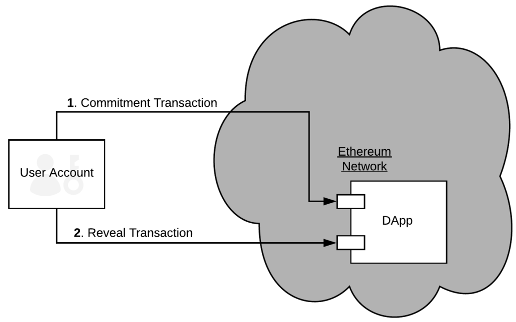

Figure 4.9: Commit and Reveal. User sends a commitment transaction with the hash of the data, After the commitment period is over, user sends her reveal transaction to the DApp revealing the information that matches the commitment.

the commitment (see Figure 4.9).

An early application of this scheme to blockchain is Namecoin, a Bitcoin-forked DApp for name services [174]. In Namecoin, a user sends a commit transaction which registers a new hidden domain name, similar to a sealed bid. Once this first transaction is confirmed, a time delay begins. After the delay, a second transaction reveals the details of the requested domain. This prevents front-running if the reveal transaction is confirmed faster than an adversarial node or miner can redo the entire process.

Commit/reveal is a two-round protocol, and aborting after the first round (early aborts) could be an issue for this (along with most multi-round cryptographic protocols). For example, in a financial exchange where the number of other orders might be in a predictable interval, an adversary can spray the sequence (*i.e.,* a price-time priority queue) with multiple committed transactions and no intention of executing

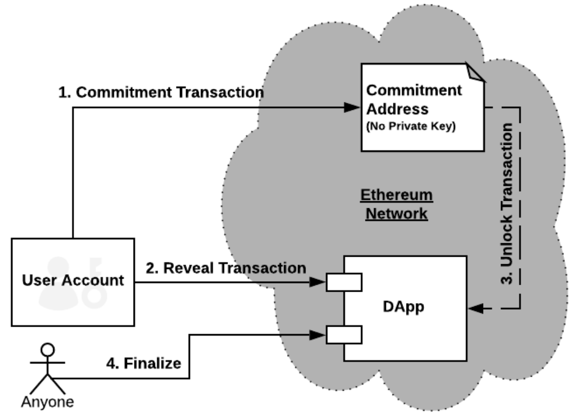

Figure 4.10: Submarine Send [49]. User generates an *Unlock* transaction from which the commitment address is retrieved using ECDSA ECRecover. 1. by funding the *commitment address*, user is committed to the transaction. 2. User sends the *reveal transaction* to the DApp, revealing the nature of the commitment transaction. 3. She broadcasts the *unlock transaction* to unlock the funds in the commitment address. 4. After the *"Auction"* is over, anyone can call *Finalize* function to finalize the process.

them all. She then only reveals the ones that result in an advantageous trade.[20] There are other ways of aborting; if payments are required but not collateralized, the aborting party can ensure that payment is not available for transfer. One mitigation to early aborts that blockchain is uniquely positioned to make is having users post a fidelity bond of a certain amount of cryptocurrency that can be automatically dispensed if they fail to fully execute committed transactions (this is used in multi-round blockchain voting [216]). Finally, we note that any multiple round protocol will have usability challenges: users must be aware that participating in the first round is not sufficient for completing their intention.

[20] This is analogous to behavior in traditional financial markets where high-frequency traders will make and cancel orders at many price points (flash orders or pinging). If they can cancel faster than someone can execute it—someone who has only seen the order and not the cancellation—then the victim reveals their price information.

Enhanced Commit/Reveal. Submarine Commitments [49, 48] extend the confidentiality of the commit and reveal, so that the commitment transaction is identical to a transaction to a newly generated Ethereum address. They initially hide the contract address being invoked, providing (3,4,5)-confidentiality during the commit phase; and they ensure that if a revealed transaction sent funds, the funds were fully collateralized at commit time and are available to the receiving smart contract. See Figure 4.10.

4.6.3 Design Practices

The final main category of mitigation is to assume front-running is unpreventable and to thus responsively redesign the functionality of the DApp to remove any benefit from it. For example, when designing a decentralized exchange, one can use a call market design instead of a time-sensitive order book [70] to side-step and disincentivize front-running. In a call market design, the arrival time of orders does not matter as they are executed in batches[21]. The call market solution pivots profitable gains that front-running miners stand to gain into fees that they collect [70], removing the financial incentive to front-run.

In the finance literature, Malinova and Park discuss front-running mitigations for blockchain-based trading platforms [208]. Instead of studying DApps, they develop an economic model where transactions, asset holdings, and traders' identities have greater transparency than in standard economic models—transparency they argue

[21] Also known as batch auctions [334]

that could be accomplished by blockchain technology. However, in their model, they assume entities can interact directly over private channels to arrange trades. They define front-running in the context of private offers, where parties might adjust their position before accepting or countering a received offer. This model is quite different from the DApp-based model we study here.

Another example in the design of ERC20 standard [331] is the allowance functionality. *approve()* function in the specification allows a second entity to be able to spend N tokens from the sender's balance. In order to change the allowance, sender must send a transaction to set the new allowance value. Using the insertion attack, attacker could front-run the new allowance transaction and spend the old value before the new value is set [273, 170], and then additionally spend the new amount at a later time. Solutions such as *decreaseApproval()/increaseApproval()* were added in updated implementations.

Baum *et al.*, expands on many of the mitigation approaches described in this section by modeling the front-running mitigation methods in decentralized finance and discuss further research directions and challenges [25]. Additionally, Heimbach *et al.* systemizes the transaction reordering manipulation mitigation schemes and analyze the full impact on the blockchain [159], and they determine that, at present, no strategy entirely satisfies all the requirements of the blockchain ecosystem.

4.6.4 Embracing Front-running

This approach has been initially advocated by a group called Flashbots as to remove the negative externalities posed by Maximal Extractable Value (MEV) [139]. MEV is the profit that miners can make by reordering transactions in a block. They propose a new type of block building mechanism, called *Flashbots Bundle*. A bundle is a set of transactions that are bundled together and submitted to the *relayer*. The bundle is not broadcasted to the network and is only visible to the relayer, in which is later shared with block proposer and builder. The block builder can then decide to include the bundle in the block or not. This method is not a solution to front-running, but it is a way to share the profit with more actors than just the block builders. Any entity can create a transaction bundle which includes their own transactions and the transactions they are front-running in the preferred order that is profitable for them. They submit this bundle to a relayer in which a portion of the profit is offered to the block builder to include the bundle in their block. This solution offers more actors to participate in MEV and share the profit –democratizing MEV–. Even though this solution does not solve the issue, it changes the dynamic of the transaction propagation and block building, and potentially reduces the network congestion load and gas price for the overall network [214].

4.6.5 Non-solutions

Some might argue that faster blockchains are immune to frontrunning attacks. This is not the case. While faster blockchains reduce the window of opportunity for front-runners, they do not eliminate it. For example, consider a blockchain with a block time of 1 second. If a front-runner can observe a transaction before it is included in a block, she can still insert a transaction in front of it. This is because the front-runner can observe the P2P network transactions and send her transaction in less than 1 second to be included in the next block.

4.7 Follow-up Work

Our paper is one of the first research papers that studies front-running attacks on the blockchains and specifically for Ethereum DApps. Since the publication of our paper in 2019, there has been a number of follow-up work that studies more complex front-running attacks and mitigation methods. In this section, we briefly discuss some of these works.

There has been countless articles and research paper analyzing the real world blockchain front-running attacks and their impact. Torres *et al.* uses our front-running attacks taxonomy (See Section 4.3) and conducts an analysis of the profitability of each type of front-running attacks in the wild. Zhou *et al.* conducted a quantitative research on the extracted value and the front-running strategies on chain [267], and in another research they showcased how some other attacks can utilize flashloans to increase the attacker's profit in DeFi applications [269].

As discussed in 4.6.2, one approach to solve frontrunning on the blockchains is to use confidentiality mechanisms to hide information from network participants. Other than the methods discussed in section 4.6.2, another promising methods is to use threshold decryption schemes [45, 29]. In this approach, utilized by projects like F3 and Anoma [358, 14, 336], every transaction is encrypted using global public key and the decryption key is distributed among a set of nodes (committee of decryptors). The nodes can then decrypt the transaction after it is finalized by the consensus mechanism and execute it. This approach is not trivial to implement, requires a trusted setup and adds significant network overhead to decrypt and execute the transactions [224]. Subsequently, FairBlock [224], proposes a solution to the bandwidth problem by using Identity Based Encryption (IBE) [41] and a novel decryption mechanism to reduce the protocol overhead. Projects including Secret Network [69] and Fairy [298] use secure enclaves, such as Intel SGX [86], to implement private mempools to hide the transaction information from the network participants. However, this approach significantly adds latency and storage overhead to the network, not to mention the security concerns regarding the security of the enclaves [183, 199, 51, 272]. Other methods also have been proposed to mitigate front-running, such as the use of verifiable delay functions (VDF) [39, 197] and multi-party computation (MPC) [361, 68, 149], which have their own challenges and limitations [224].

Another follow-up research that has received a lot of attention is the study of Maximum Extractable Value (MEV) [100] and its impact on the blockchain ecosystem (See section 4.6.4). Many papers conducted quantified research on the extractable value by Flashbots [139] and similar MEV relays [342, 195] using different data gathering methods. In contrast, many other researchers focused on the mitigation techniques and how to minimize the available value to be extracted [362, 350]. There are discussions regarding the centralization factor MEV relays bring to the blockchain ecosystem, and with that the chances of transaction censorship. Wahrstätter *et al.* [332] conducted a research formalizing and analyzing the security impact of blockchain censorship. They discuss the impact of Proposer Builder Separation (PBS) [341] on security and censorship resistance of Ethereum network. Additionally, they show that if the censoring validators make up more than 50% of the network participants, no PoS (and PoW) protocol can achieve censorship-resilience [332]. Furthermore, Heimbach *et al.* [158] dive deeper in PBS theory and its impact on network centralization and transaction censorship. Zhang *et al.* [360] took an interesting approach to utilize MEV to counterattack smart contract exploits, by introducing `STING`, which attempts to front-run the attacker's transaction.

4.8 Concluding Remarks

Following the theme of this dissertation, blockchain permissionless and transparent aspects have brought new challenges to the security of the system. Front-running in the traditional financial markets are only feasible to the entities in the flow of information, however, in blockchain, the information is available to everyone. In this chapter, we studied the front-running attacks on the blockchain and Ethereum DApps. We discussed the different types of front-running attacks and their impact on the blockchain ecosystem. We also discussed the different mitigation methods and how they can be applied to prevent or reduce the impact of front-running attacks.

Front-running is a pervasive issue in Ethereum DApps. DApp developers don't necessarily have the mindset to design DApps with front-running in mind. This is an attempt to bring forward the subject and increase awareness of these type of attacks. While some DApp-level application logic could be built to mitigate these attacks, its ubiquity across different DApp categories suggests mitigations at the blockchain-level would perhaps be more effective. We highlight this as an important research area.

4.8.1 Ethics and Legality

The underlying technology of Ethereum is not designed to prevent front-running attacks. However, many researchers and industry leaders have been discussing the issue and possible solutions for a while. As we discussed in 4.2.2, in traditional markets, the idea of front-running orders took more than 3 decades to fully understand and implement legal solutions to prevent it. When it comes to blockchain, the technology is still new and people are still figuring out all the possible attacks and how to stop them. This research does not dive deep into the ethical problems of MEV [214], but it does bring up a big question: in a system where people are paid to act honestly and MEV lets them seek extra profits by extracting value from other users, is this still honest and ethical behavior? For example, MEV can affect many users by changing the order price after a user sends a transaction but before it goes through. In traditional rules, this is seen as a kind of market manipulation or insider trading and is illegal. So, it's important to think about how these actions and rules apply in the blockchain world and how the technical nuances of the technologies we use may enable such unforeseen consequences.

Chapter 5

Oracles: from Ground Truth to Market Manipulation

This chapter is based on the paper "Oracles; from the ground truth to market manipulation" [123] published at 3rd ACM Conference on Advances in Financial Technologies. This paper was supervised by Jeremy Clark, and co-authored by Mehdi Salehi and Wanyun Catherine Gu.

5.1 Introduction

In the previous chapter, we discussed the front-running attacks that can change the underlying information a user's transaction is based on and change the state of the execution prior to the transaction. In this chapter, we focus on the source of truth and how the information flow can be manipulated by different actors depending on the *oracle* technical design. These attacks vary from abuse of the trust to technical nuances of the design that can be manipulated by malicious network participants. We start by introducing the oracle problem and the different oracle designs. We then present a modular framework for oracle designs and discuss the different attacks that can be performed on each module. We conclude by presenting a taxonomy of oracle designs and discuss the different oracle designs in the context of the taxonomy.

5.2 Background and Motivation

With billions of dollars at stake, decentralized networks are prone to attacks. It is essential that the smart contracts, which govern how systems are run on these networks, are executed correctly. Public blockchains, like Ethereum, ensure the correct execution of smart contract code by taking the consensus of a large, open network of nodes operating the Ethereum software. For consensus to form, many nodes need to make decisions based on the exact same input data. Hypothetically, if a decision requires nodes to fetch data or use a service provider outside the blockchain, there can be no guarantee that every node in a global network has the same access and view of this external source. For this reason, blockchains only execute on internal sources: data and code provided in a current transaction, or past data and code already stored on the blockchain.

Many potential decentralized applications seem very natural until the designer hits the 'oracle problem' and realizes an interface to the external world is required. An oracle is a solution to this problem. It is a service that feeds off-chain data into on-chain storage. The trust model of oracles vary—some data comes with cryptographic certification while other data is assumed to be true based on trusting the oracle, or a set of oracles. Oracle-supplied data cannot easily be changed or removed once finalized on-chain, allowing disputes over data accuracy to be based on a public record. Leveraging this immutability is one approach to incentivizing oracles to post truthful information.

We aim to construct a systematization of knowledge (SoK) of implementation choices for oracles, facilitated by breaking down the operation of an oracle into a set of modules. For each module, we explore potential system vulnerabilities and discuss attack vectors. We also aim to categorize all the significant oracle proposals of different projects within a taxonomy we propose. The goal of this research is to help the reader better understand the system design for oracles across different use cases and implementations.

5.3 Preliminaries

5.3.1 The Oracle Problem.

Smart contracts cannot access external resources (*e.g.,* a website or an online database) to fetch data that resides outside the blockchain (*e.g.,* a price quote of an asset). External data needs to be relayed to smart contracts with an oracle. An *oracle* is a bridge or gateway that connects the off-chain real world knowledge and the on-chain blockchain network. The 'oracle problem' [63] describes the limitation with which the types of applications that can execute solely within a fully decentralized, adversarial environment like Ethereum. Generally speaking, a public blockchain environment is chosen to avoid dependencies on a single (or a small set) of trusted parties. One of the first oracle implementations used a smart contract in the form of a database (*i.e.,* mapping[1]) and was updated by a trusted entity known as the `owner`. More modern oracle updating methods use consensus protocol with multiple data feeds or polling techniques based on the 'wisdom of the crowd'. The data reported by an oracle will always introduce a time lag from the data source and more complex polling methods generally imply longer latency.

5.3.2 Trusted Third Parties.

A natural question for smart contract developers to ask is: if you trust the oracle, why not just have it compute everything? How is trusting the oracle, different from trusting a third party to provide the information? There are a few answers to this question: (1) there may be benefits to minimizing the centralized trust (*i.e.,* to just providing data instead of full execution), (2) there are widely trusted organizations and institutes—convincing one to operate an oracle service is a much lower technical ask than convincing one to operate a complete platform, and (3) if a data source becomes untrustworthy, it may require less effort to switch oracles than to redeploy

[1] A Solidity `mapping` is simply a key-value database stored on a smart contract.

the system. These points are not meant to be exhaustive, but rather to illustrate that there are many reasons why a developer may choose to implement their own oracle solution or distribute trust by using an oracle network instead of one trusted third party.

5.3.3 Methodology.

We found papers and other resources by examining the proceedings of top ranked security, cryptography, and blockchain venues; attending blockchain-focused community events; and leveraging our expertise and experience. Our inputs include academic papers, industry whitepapers, blog and social media posts, and talks at industry conferences on blockchain technology, Ethereum, and decentralized finance (DeFi).

5.3.4 Oracle Use-Cases.

Oracles have been proposed for a wide variety of applications. Based on our reading, most of the use-cases fall into one of the main categories below.

- **Stablecoins** [71, 223, 256, 152, 207] and **synthetic assets** [285] require the exchange rate between the asset they are price-targeting and the price of an on-chain source of collateral.

- **Derivatives** [120, 33, 301] and **prediction markets** [70, 257] require external prices or event outcomes to settle on-chain contracts.

- **Provenance systems** [**?**, 319] require tracking information of real world assets like gold, diamonds, mechanical parts, and shipments.

- **Identity** [180, 209] and other on-chain reputation systems require knowledge of governmental records to establish identities.

- **Randomness** [62] can only be produced deterministically on a blockchain. In order to use any non-deterministic random number, an external oracle is needed to feed the randomness into the smart contract. **Lotteries** [264] and **games** [131] are examples. Additionally, cryptographic tools like verifiable random functions (VRF) [218, 147] and verifiable delay functions (VDFs) [53, 40] can mitigate, respectively, any predictability or manipulability in the randomness.

- **Decentralized exchanges** can use prices from an external oracle to set parameters. On-chain market makers [162] uses such prices to minimize the deviation from the external market prices and tailor the pricing function. Additionally, some use oracles to provide sufficient liquidity near the mid-market price for more efficient automated market making [108, 353, 270].

- **Dynamic non-fungible tokens (NFTs)** [37] are crypto-collectables that can be minted, burned, or updated based on external data. For example, sports trading cards which depends on the real-time performance of a player.

5.4 Modular Work Flow

For our main contribution, we deconstruct how an oracle operates into several modules that generally operate sequentially (but in some solutions, certain steps are skipped) and then we study each module one-by-one. An overview of the work flow is as follows:

5.4.1 **Ground Truth**: The goal of the oracle system is to relay the ground truth (*i.e.,* the real true data) to the requester of the data.

5.4.2 **Data Sources**: Data Sources are entities that store or measure a representation of the ground truth. There are a diverse set of data sources: databases, hardware sensors, humans, other smart contracts, *etc.*

5.4.3 **Data Feeders**: Data feeders report off-chain data sources to an on-chain oracle system. In order to incentivize truthful data reporting, an oracle system can introduce a mechanism to select data feeders from a collection of available data providers. The incentive mechanism can be collateral-based, such as staking, or reputation-based to find a reliable set of data feeders for each round of selection.

5.4.4 **Selection of Data Feeders**: The process of determining which data feeders should be used in an oracle system can be categorized into two main types: centralized and decentralized selection.

5.4.5 **Aggregation**: When data is submitted by multiple data feeders, the final representation of the data is an aggregation of each data feeder's input. The aggregation method can be random selection or algorithmic rule-based, such as using weighted average (the mean) or majority opinion. The design of the aggregation method is one of the most important aspects of an oracle system, as intentional manipulation or unintentional errors during the aggregation process can result in untruthful data reporting by the oracle system.

5.4.6 **Dispute Phase**: Some oracle designs allow for a dispute phase as a countermeasure to oracle manipulation. The dispute phase might correct submitted data or punish untruthful data feeders. The dispute phase might also introduce further latency.

The steps above are visualized in Figure 5.1. Next we dive deeper into the modular workflow by trying to further define each module. As appropriate, we also discuss feasible attacks on the modules and possible mitigation measures.

5.4.1 Ground Truth

While not a module itself, ground truth is the initial input to an oracle system. Oracle designers cannot solve basic philosophical questions like *what is truth?*, However, it has to be understood (i) what the data actually represents and (ii) if it is reliable. Data is sometimes sensitive to small details. Consider a volatility statistic for a financial asset: basics like which volatility measure is being used over what precise time period are obvious, but smaller things like the tick size of the market generating the prices could be relevant [138]. When data is aggregated from multiple sources,

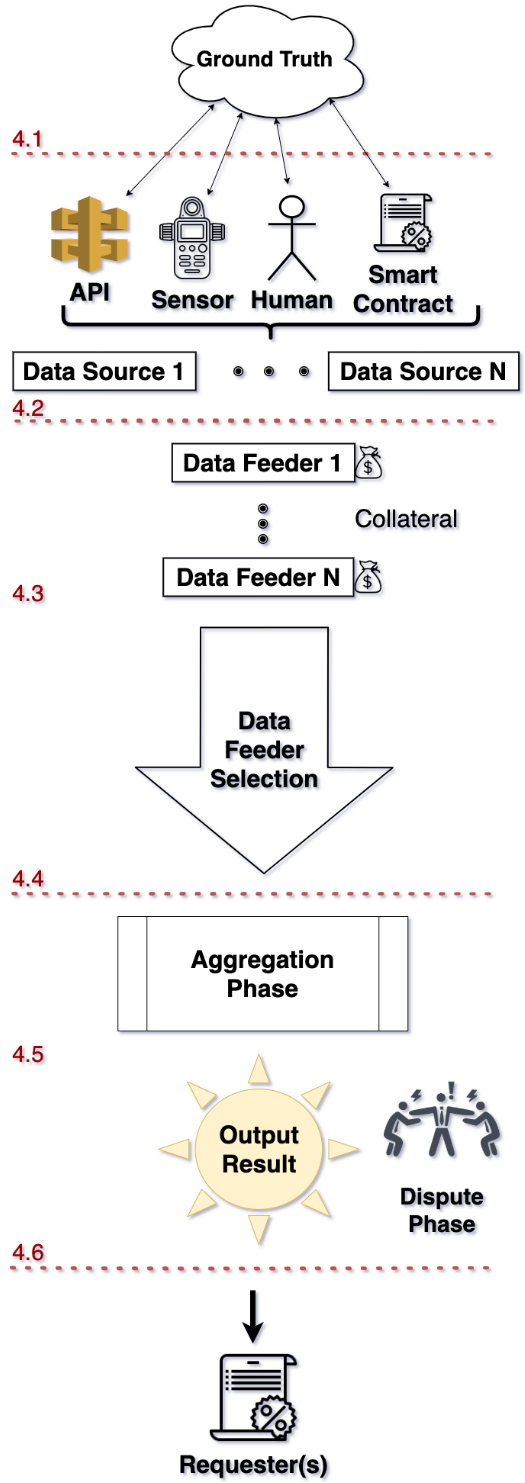

Figure 5.1: A Visualization of Our Oracle Workflow

minor differences in what is being represented (called *semantic heterogeneity*) can lead to deviations between values [206, 348, 153].

While oracle systems will attempt to solve the issue of malicious participants who misreport the ground truth, it does not address the fundamental question of whether the ground truth itself is reliable. Some philosophers argue truth is observed, and observations require a 'web of beliefs' that is subject to error (for its consequences in security, see [161]). Reliability is judged by the assumptions made about the data source, described next.

5.4.2 Data Sources

Data Sources are defined here as passive entities that store and measure the representation of the ground truth. Common types of data sources include `databases`, `sensors`, `humans`, `smart contracts`, or a combination of them. Depending on how data sources gather and retrieve the ground truth, different attack types arise. Using a hybrid of data sources (if possible) could reduce the reliability on a single point of input. We describe each common type and their security considerations.

5.4.2.1 Humans.

A human may provide the requested data, either by direct observation or by indirectly relaying data from another data source. Humans are prone to errors which is the main risk of this data source. Human errors include how the data is retrieved, how the data collector interprets the truth, and if data is relayed from a reliable source.

Researchers have categorized human errors into the following three types (from least to most probable): very simple tasks, routine tasks, and complicated non-routine tasks [203]. An example for each category is, respectably, reading Bitcoin's exchange rate from an unverified source, inputting the data into the system, and configuring the oracle system.

Humans may also act maliciously and deliberately report wrong data when they perceive it will benefit them. As we will see in further modules, a robust oracle system will use incentives and disputes to promote truthful statements.

5.4.2.2 Sensors.

Sensors are electronic devices that collect raw data from the outside world and make it available to other devices. The data source may use more than one sensor to obtain the desired data. One example from traditional finance is the weather derivative, first introduced by the Chicago Mercantile Exchange (CME) [229]. These instruments use weather data provided by trusted institutions, such as the National Climate Data Center,[2] which collects weather data through a network of sensors.

Provenance is a highly cited application of blockchain, where products are tagged and traced throughout the supply chain, including transportation, for management and/or certification [319, 225, 354]. The tags could be visual (Barcode) or electronic (RFID). A host of attacks on RFID have been proposed outside of blockchain oracles [8]. Blockchain technology does not solve some important trust issues: ensuring

[2] `https://www.ncdc.noaa.gov/`

the proper tag is affixed to the proper product, each product has one tag, each tag is affixed to only one product, and tags cannot be transferred between products. This is called the *stapling* problem [280].

Sensors can produce noisy data or malfunction. The hardware of a sensor can also be modified when remote or physical access is unauthenticated (or weakly authenticated as many sensors are constrained devices). Probably the highest profile sensor attack (outside of blockchain) is Stuxnet [193]—malware that manipulated the vibration sensors, the valve control sensors, and the rotor speed sensors of Iran's nuclear centrifuges, causing the system to quietly fail [192].

5.4.2.3 Databases and APIs.

The most common mechanism used by software to fetch data is to use an API (Application Programming Interfaces) to obtain the data directly from a centralized database. A database is a set of tables that collect system events, while the API is an interface with the database. For example, a financial exchange keeps track of information in a database about every trade that has been executed. A data source that needs the daily traded volume of an asset could use the appropriate API of the exchange's database to extract the data from the related table in the database.

An active attacker can attack the system from two points. Modifying the data at rest in the database, or modifying the data in transit before and after the API call.

5.4.2.4 Smart Contracts.

Smart contract could be used as a data source similar to a database. Decentralized finance (DeFi) applications on Ethereum include decentralized exchange services like Uniswap [5], or other oracles that operate on-chain. For instance API3 oracle [30] uses other on-chain oracles, called *dAPIs*, as their data source. These oracles are whitelisted through voting by API3 token holders.

Automated Market Makers (AMMs) [335] are an on-chain alternative to centralized exchanges. Liquidity providers collateralize the contract with an equally valued volume of two types of cryptoassets. A mathematical rule governs how many assets of the one type are needed to purchase assets of the other. A well-known example of such mathematical rule is the *Constant Function Market Makers* (CFMM) to calculate the exchange rates of tokens in a single trade [290]. The idea behind AMM was first raised by Hanson's logarithmic market scoring rule (LMSR) for prediction markets [155]. A class of DeFi projects (*e.g.,* Uniswap [5, 6] and Balancer [22]) uses CFMM to automate their market-making process. One of the utilizations of AMM is the ability to measure the price of an asset in a fully decentralized way, which addresses the *pricing oracle problem* [11].

One potential attack vector to the auto price discovery mechanism in an AMM is to manipulate prices provided by an algorithm, since the algorithmic rules used by an AMM is written in the smart contract and therefore how prices are quoted by the AMM can be calculated in advance. a real case example on *bZx* is described in Section 5.5.3.2. In addition to market manipulation, *smart contract vulnerabilities* [15, 66] could possibly be used to influence the data coming from the oracle, which we will discuss more in section 5.5.3.

5.4.3 Data Feeders

Data feeders are entities who gather and report the data from a data source (Section 5.4.2) to the oracle system. A common configuration consists of an *external feeder* which draws from off-chain data sources and deposit the data to an on-chain module. In case the data source is already on the blockchain, the data feeder step can be skipped.

It is not common to assume data feeders are fully honest, however a variety of threat models exist. Generally, this module will not attempt to determine if the data has been falsified (the data selection (Section 5.4.4), data aggregation (Section 5.4.5) and dispute phase (Section 5.4.6) modules will deal with this issue); rather it will consist of tunneling the data through the feeder with some useful security provisions. We discuss most important security provisions to achieve data integrity, confidentiality, and non-repudiation on any specific data.

5.4.3.1 Source Authentication.

Data integrity can be enhanced by authenticating the source of the data and ensuring message integrity is preserved. It is sufficient to have the source sign the data, assuming the source's true signature verification (*i.e.,* public) key is known to the recipient of the data. This is most appropriate for sources like humans and sensors (although sensors may use a lightweight cryptographic alternative to expensive digital signatures [286]).

Databases, websites, and APIs typically support many cryptographic protocols, including the popular HTTPS (HTTP over SSL/TLS) which adds server authentication and message integrity to HTTP data [74]. However, HTTPS alone is typically not sufficient, as the message integrity it provides can only be verified by a client that connects to the server and engages in an interactive handshake protocol. This client cannot, for example, produce a transcript of what occurred and show it to a third party (*e.g.,* a smart contract on Ethereum) as proof that the message was not modified. To turn HTTPS data into signed data (or something similar), a trusted third party can vouch that the data is as received. TLS notary [320] and DECO [357] offer solutions that attest for the authenticity of HTTPS data. Town Crier [356] uses Trusted Execution Environments (TEE) like Intel SGX [86] to push the trust assumption onto TEE technology and, ultimately, the chip manufacturer.

5.4.3.2 Confidentiality.

For many smart contracts that rely on oracles, the final data is made transparent (*e.g.,* prices, weather, event outcomes). In a few cases, oracles feed data that is private (*e.g.,* identities, supply chain information) and the contract enforces an access structure of which entities under which circumstances can access it [209].

Confidentiality might also be temporary. Given the fact that information submitted to the mempool is public, there is a natural risk on the oracle system that a data feeder uses another data feeder's information to self-report to the system. This form of collusion between data feeders is called *mirroring attack [112]* in computer security literature. The data feeders are willing to freeload another data feeder's response to minimize their cost of data provision. They will also be confident that their data will not be an outlier and be penalized. To mitigate the risk of mirroring attacks, the oracle designer should consider mechanisms that ensure the confidentiality of the data sent by the data feeders. A popular technique to achieve confidentiality is to use a commitment scheme [43]. In a commitment scheme, each data feeder should send a commitment of the plain data as an encrypted message to the receiver. Later, the sender can reveal the original plain data and verify its authenticity using the commitment.

5.4.3.3 Non-Repudiation.

A non-repudiation mechanism assures that a party cannot deny the sender's proposal after being submitted to the system. Oracle systems might rely on cryptographic signature schemes to eliminate the risk of in-transit corruption and to create irrefutable evidence of the data being provided by a source, for use in the dispute phase (Section 5.4.6) as needed.

5.4.4 Selection of Data Feeders

In order to ensure correct data is fed into the system, the design must select legitimate data feeders and weed out less qualified and malicious participants. In a

Category	*Example*	No Trusted Third Party	Low latency	Resilient to Sybil Attacks	Resilient to Targetted DoS Attacks	Incentives are Endogenous
Centralized	`Maker V1 Oracle`		●	●		
Voting	`Maker V2 Oracle`	●			●	
Staking	`Chainlink`, `ASTRAEA`	●	●	○	●	●

Table 5.1: Evaluation Framework on selection of data feeders. For details see Section 5.4.4.4

non-adversarial environment, the design might aggregate all the incoming data without any selection, skipping this step.

The earliest designs for oracle systems, such as Oraclizeit [31] and PriceGeth [120], were designed using just one single data feeder; however, to improve data quality and the degree of decentralization, more complex oracle systems such as ChainLink [112] involves selecting qualified data feeders to aggregate an output that is expected to be more representative of the ground truth.

This process can be categorized into two main types: centralized and decentralized selection, with decentralized selection having multiple approaches through voting and staking. Centralized selection and decentralized selection through voting, create an allowlist of legitimate data feeders, in contrast to selecting based on the algorithmic criteria in decentralized selection through staking.

5.4.4.1 Centralized (Allowlist) Selection.

A centralized selection is a permissioned approach where a centralized entity selects a number of data feeders directly without the involvement of other participants in the network. A centralized selection is analogous to having an allowlist for authorized data feeds (*e.g.,* Maker Oracle V1 [207]). Compared to a decentralized approach, centralized selection is fast and direct. The trust footprint on the central entity is large: it must solely select legitimate data feeders and also have high availability to update the allowlist as needed.

5.4.4.2 Decentralized (Allowlist) Selection through Voting.

By decentralizing the selection process, the goal is to distribute the trust from a single entity to a collective decentralized governance. Voting distributes trust and provides a degree of robustness against entities failing to participate, however it adds latency and introduces the threat that an actor can accumulate voting rights to sway the vote [204], or even to do distrust and destroy the system (*e.g.,* Goldfinger attack [189]).

For instance, in Maker V2 oracle [207], the selection of the data feeders is done through a decentralized governance process [152]. MKR[3] token holders vote on the number of authorized data feeders and who these data feeders can be [78].

Note that sometimes voting processes can provide the illusion of decentralization while not being much different from a centralized process in practice. To illustrate,

[3] MakerDAO Governance Token

consider a project with a governance token, in which most tokens are held by a few individuals where the project leaders advocate for their preferences and there is no established venue for dissenting opinions. If voters only inform themselves from one source of information, that source becomes a de facto centralized decision maker.

5.4.4.3 Decentralized Selection through Staking.

Like voting, staking attempts to utilize a token to align the incentives of the participants with the current functioning of the system. Mechanically, it works different: data feeders post collateral against the data they provide. In the dispute phase 5.4.6, any malicious data feeders will be punished by losing a portion or all of their collateral (called *slashing*). Even without slashing, the collateral amount acts as a barrier to entry for participants and rate-limits participant.

The stake can be both in token value and reputation of the data feeder. As an example, in Chainlink [112] protocol has a reputation contract that keeps track of the accuracy of data reporting of different feeders. The `ExplicitStaking` module in Chainlink 2.0 defines the number of Link tokens each oracle node must stake to become a data feeder, while the service agreement of the Chainlink oracle defines the circumstances in which a node's stake will be slashed [250]. Put together, the incentives for selected data feeders to act honestly are avoiding reputational loss, avoiding loss of stake and penalty fees, and maintaining good standing for future income. In terms of selection, the data selection module forms a leaderboard, based on collateral and reputation, to select the highest ranked data feeders from all available feeders.

Another approach, introduced by ASTRAEA [7], uses a combination of game theory and collateralization between different actors in the system (Voters and Certifiers) to achieve equilibrium on what the final data should be.

A staking-based selection module avoids a central trusted third party, but it can add latency for adding/remove data feeds and other adjustments. It is also open to sybil attacks by design, while working to ensure these attacks have a significant cost for the adversary.

One challenge for designing a staking mechanism is setting a high enough punishment (slashing) mechanism to thwart malicious actions. Projects like UMA [323], another smart contract oracle design, dynamically adjust staked collateral needed for each round to ensure that *Cost of Corruption* (CoC) is higher than the projected *Profit from Corruption* (PfC). Profit from Corruption is defined by the data requester, in which UMA contracts require higher collateral to finalize the data from the data feeders. It is also important that participants are incentivized to file correct disputes—ones that will ultimately lead to identifying misbehavior. If disputes are filed on-chain, the disputer will have to pay gas costs that need to be ultimately reimbursed by the resolution process.

Decentralized selection is done by the holders of some scarce token, typically a governance token specific to the oracle service. The simplest decentralized mechanism to hold a vote amongst token holders, who are indirectly incentivized (we call this an *exogenous incentive*) cast informed votes since they hold a token tied to the success of the system (*e.g.,* TruthCoin [302]). In a staking system, token holders are directly incentivized (an *endogenous incentive*) to vote 'correctly' (this remains to be defined but assume for now it means they vote in a way that will not be disputed) by posting some amount of their tokens as a fidelity bond. Stakers stand to be rewarded with new tokens and/or penalized (collateral slashed) depending on the performance of the data feeders they vote for.

Additionally, a protocol could introduce a random selection within the data feeders to decrease the chance of sybil attacks. As an example Band Protocol [266], chooses a random validator from top 100 staked participants for their oracle system.

Another approach used by Tellor oracle [309] is a simple Proof of Work (PoW) algorithm for each round of data. The first 5 miners to submit their desired data alongside the solution to the mining puzzle are selected as the data feeders of the round. The selection is based on the hash power of each data feeder and randomness nature of proof of work consensus.

5.4.4.4 Evaluation Framework on the Data Feeders Selection

To compare designs for data feeder selection, we provide an evaluation framework. The definition of each evaluation criteria (*i.e.,* column of the table) follows, specifying what it means to receive a full dot (•), partial dot (◦) or to not receive a dot.

No Trusted Third Party. A selection process that is distributed or decentralized amongst several equally-powerful entities earns a full dot (•). A process that relies on a single entity for critical functions is not awarded a dot.

The voting and staking processes are decentralized amongst multiple token holders (●). As the name implies, the centralized process uses a trusted third party (no dot).

Low Latency. A selection process that can move from proposal to finality within a single transaction is awarded a full dot (●). A process that requires multiple rounds of communication or communication amongst several entities is not awarded a dot.

The centralized process can make selection decisions unilaterally (●). The voting process involves a round of communication with all the participants (no dot). The staking process draws feeders unilaterally from an established leaderboard (●).

Resilient to Sybil Attacks. A selection process that only allows unique feeders to participate is awarded a full dot (●). The evaluation does not consider what specific method is used to determine entities are unique but assumes it works reasonably well (not strictly infallible). A process that is open to multiple fake feeders controlled by the same adversary is awarded a partial dot (○) if each additional feeder created by the adversary has a material financial cost. If there is no material cost to creating additional fake feeders, the process receives no dot.

The centralized process manages an allowlist based on real world reputations. We assume this reasonably prevents sybils (●). The staking process admits sybils but deters them by requiring staked tokens for each, which is costly (○). The voting process does not deter sybils from entering the election but relies instead on the voting process to not select them (no dot).

Resilient to Targeted Denial of Service Attacks. A selection process that only halts when multiple entities to go offline or fail is awarded a full dot (●). If critical *functionalities cannot be performed with the failure of a single entity, but the basic selection process can proceed, it is awarded a partial dot (○). If the process can be fully halted by the failure of a single entity, it is awarded no dot.*

The voting and staking processes can proceed until enough honest participants fail that a dishonest majority remains (●). By contrast, a failure with the central entity in a centralized process can prevent critical functionalities, like updating the allowlist (○).

Incentives are Endogenous. Every selection process should have the ability to remove untruthful feeders. Some selection processes might go beyond this and incentivize feeders to provide truthful information. Processes are awarded a full dot (●) if the awards/punishments can be realized by the selection process itself. If the selection process relies only on external incentives (e.g., damage to reputation), it is awarded no dot. The evaluation does not consider how information is determined to be truthful or not. Endogenous means the design is simpler but does not imply it is more secure (cf. [141]).

The staking process requires feeders to post collateral that can be taken (*i.e.,* slashed) for malicious behavior (●). Centralized and voting processes do not use internal incentives (no dot).

5.4.5 Aggregation of Data Feeds

Aggregation is the process of synthesizing the selected data feeds into one single output. The quality of the output depends on the data feed selection (see Section 5.4.4) and the aggregation process used. To highlight the importance of designing an aggregation method correctly, consider the case of Synthetix, a trading platform [301] that used the *average* (or *mean*) of two data feeders as their aggregation method. An attacker leveraged this to manipulate one of the two feeders by inflating the real price by 1000x. Mean aggregation is highly sensitive to outlier data and the attack resulted in Synthetix's loss of several million dollars [300].

5.4.5.1 Statistical Measures.

The three core statistics for aggregation are mean, median and mode. Many oracle systems use the median as the aggregated output, by selecting the middle entry of a list of ordinal data inputs. Unlike the mean, the median is not skewed by outliers, although it assumes the inputs have an appropriate statistical distribution where the median is a representative statistic for the underlying ground-truth value. For example, if we believe data from the feeders is normally distributed with possible outliers, the median is appropriate. However, if we believe it is bi-modally distributed, then discretizing and computing the mode (most common value) of the data is more appropriate. The mode is useful for non-numeric data (and nominal numbers). An approximation to the mode is picking a data input at random, however access to randomness from a smart contract is a well-documented challenge [64, 53, 62]. Oracle projects like Chainlink do not prescribe a fixed aggregation method and let the data requesters select one.

To improve the quality of simple statistics such as the median and the mode, weights can be applied in the calculation. For instance, to mitigate manipulation of price data, one can choose to use *time-weighted average price* (TWAP) [324], or liquidity volume, or both [6]. Typically, the liquidity and trading volume of a market correlates with the quality of the price data. To illustrate, Uniswap V2 uses TWAP over several blocks (*e.g.,* mean price in the last 10 blocks) to reduce the possibility of market manipulation in a single block (*e.g.,* via flash loans [268]). In Uniswap V3, TWAP is optimized for more detailed queries including the liquidity volume and allowing users to compute the geometric mean TWAP [6].

5.4.5.2 Stale Data.

Some use cases require frequent updates to data, such as weather data and asset prices. Stale data can be seen as valid data and pass the selection criteria, but it will reduce the aggregated data quality. Projects like Chainlink rank feeders based on historic timeliness. A naive approach ignores this issue and always uses the last submitted data of a data feeder even if the data feeder has not updated its price for some specific period. This approach is problematic if the underlying data is expected to change frequently. An example occurred on Black Thursday 2020 [343] to MakerDao when Maker's data feeders could not update their feeds because of very high network congestion. After a significant delay in time, feeds were updated. The price had shifted by a large amount and the reported data jumped, leading to sudden, massive liquidations that were not adequately auctioned off.

5.4.6 Dispute Phase

The dispute phase is used to safeguard the quality of the final output and give the stakeholders a chance to mitigate inclusion of wrong data. Dispute resolution can be an independent module after the aggregation phase, or it can be implemented at any other oracle module (*e.g.,* at the end of every aggregation 5.4.5 or data feeder selection 5.4.4). Most oracle systems do dispute resolution internally, but market specialization has produced firms that provide outsourced dispute resolution as service (*e.g.,* Kleros [194]). To systemize the landscape, we first distinguish between systems that aim to detect (and remove) bad data providers and systems that vet the data itself. We then iterate how data is determined to be valid or invalid for the purposes of a dispute. Finally, we illustrate the consequences of a successful dispute: what happens to the disputed data and what happens to its provider.

5.4.6.1 Provider-level and Data-level Vetting

Dispute resolution can be *provider-oriented* or data-oriented. Under a *provider-oriented* regime, the focus is on selecting honest data providers and using disputes to remove data providers from serving as oracles in the future. In the optimistic case that providers are honest, oracle data is available immediately, however if an honest provider is corrupted, it will have a window of opportunity to provide malicious data before being excluded. One illustration of a provider-oriented system is operating a centralized allowlist of data providers (*e.g.,* MakerDAO v2) where providers can be removed. Chainlink [112] strives to decentralize this functionality, where a

	No Disputes	Disputed
Data is correct	Correct	False Positive
Data is incorrect	False Negative	Correct

Table 5.2: Dispute phase types of errors

reputation-based leaderboard replaces the allowlist.

In a data-oriented regime, the focus is vetting the data itself. This can result in a slower system as oracle data is staged for a dispute period before it is finalized, however it can also correct false data (not merely remove the corrupted data feeder from future submissions). One illustration of a data-oriented system is Tellor [309, 85], where data is staged for 24 hours before finalization. If it is disputed, a period of up to 7 days is implemented to resolve the dispute. It is also possible that a system allows the resolution itself to be further disputed with one or more additional rounds. In Augur [257] for instance, the dispute step may happen in one round (takes maximum 1 day) or may contain other rounds of disputes that can last more than 7 days.

5.4.6.2 Determining the Truth

In the optimistic case, an oracle system will feed and finalize truthful data, while disputes enable recourse for incorrect data. However, disputes also introduce the possibility of two types of errors.

Dispute resolution in oracle systems focus on false positives. Incentivizing the discovery of false positives is present in some staking-based systems, however false negatives are not otherwise dealt with. In order to resolve a false positive, correct data must be used as a reference but, of course, if correct data is available as a reference, then it could replace the entire oracle system. That leaves two reasons for why an oracle system might still exist: (a) the reference for correct data is too expensive to consult on a regular basis, or (b) there is no reference for correct data, and it must be approximated.

If feeders are placed on an allowlist by a trusted party, disputes could be filed with the trusted party and manually verified. As far as we know, this is the only example of (a), although (a) is the basis for other blockchain-based dispute resolution protocols like optimistic roll-ups [173]. The rest of the truth discovery mechanisms are based on (b) approximating the truth.

A *statistical approach* is selecting, from a set of values proposed by different feeders, the median of the values (*e.g.,* appropriately distributed continuous numerical data) or the mode (*e.g.,* non-continuous or non-numerical data). It is possible to augment this approach by having feeders *stake* collateral in some cryptocurrency (*e.g.,* a governance token for the oracle project), and this collateral is taken (*slashed*) from the feeder if their data deviates from the median by some threshold. If the amount slashed is paid, in part or in full, to the entity that filed and/or supported a dispute on the data, this incentivizes feeders to help reduce false negative errors in addition to false positives. One challenge is setting an acceptable threshold for slashing. A large threshold tolerates moderately incorrect data without punishment, while a small threshold could punish data feeders that are generally honest but faulty, slow, or reporting on highly volatile data.

If a governance token exists for the oracle project, a related approach is to introduce *voting* on disputed data by any token holder, and not limit the decision to just the feeders. In Augur [257] and ASTRAEA [7], disputers vote to change the tentative outcome because they believe that outcome is false. Voting occurs over a window of time which extends the time to resolve disputes. By comparison, statistical mechanisms can be applied automatically and nearly instantly after the data is aggregated. However, voting incorporates human judgement which might produce better outcomes in nuanced situations.

One final truth discovery mechanism is *arbitrage* which is applicable in the narrow category of exchange rates between two on-chain tokens. This can be illustrated by the NEST oracle [239] where data feeders assert the correct exchange rate between two tokens by offering a minimum amount of both tokens at this rate (*e.g.*, 10 ETH and 39,000 USDT for a rate of ETH/USDT = 3900). If the rate is incorrect, other participants will be given an arbitrage opportunity to buy/sell ETH at this rate, an action that can correct the price. This is very similar to drawing a price from an on-chain exchange, like Uniswap, and suffers from the same issue: an adversary can manipulate the oracle price by spending money. It is secure when the *Cost of Corruption* (CoC) is greater than the *Profit from Corruption* (PfC), however PfC can never be adequately accounted for because profit can come from extraneous (extra-Ethereum) factors [141]. The UMA [191] oracle system has data feeders provide their own PfC estimates for the data they provide.

5.4.6.3 Consequences for Incorrect Data

We now consider the consequences for disputed data that has been determined to be incorrect. In provider-oriented dispute resolution, incorrect data has consequences for the data feeder (see next subsection) but not the data itself. By the time the dispute is resolved, it is *too late* to change the data itself.

In data-oriented dispute resolution, data that has been deemed incorrect can either be *reverted* or *corrected*. Reversion means the outcome result will be annulled, and the system should start from scratch to obtain new data, while corrected data will reflect a new undisputed value. The difference between the two is essentially in the complexity of the dispute resolution system. For reversion, a collective decision is taken to accept or reject data — a binary option that is known in advanced. By contrast, correcting data requires new data to be proposed and then a collective decision to be made on all the proposals which is more complex but does not avoid rerunning the oracle workflow.

These differences also impact *finality*: when should oracle data be considered usable? Dispute periods, re-running the workflow, and allowing resolved disputes to be further disputed can all introduce delays. To illustrate, consider Augur [257] which implements a prediction market on binary events. Any observer with an objection to a tentative outcome can start a dispute round by staking REP (Augur's native token) on the opposite outcome. Dispute windows are 24 hours and then extended to 7 days for disputes on disputes. If the total staked amount exceeds 2.5% of all REP tokens, the market enters a 60-day settlement phase called a fork window when all

REP holders are obliged to stake on the final outcome.

5.4.6.4 Consequences for Data Feeder

If data has been deemed incorrect through disputes or rejected for being an outlier, the feeder who provided the data might face consequences like being banned, slashed, or suffering reputational loss. It is also possible that there is *no consequence* for the feeder other than the data being discarded. For example, in a sensor network, results from faulty sensors could have their data filtered out but continue to contribute data in expectation that they will be repaired in the future.

In oracle designs based on allowlists, a feeder could be *banned* or temporarily suspended for providing incorrect data. For dispute resolution based on staking, a feed could suffer *economic loss* by having their stake taken from them. It is important to reiterate that this economic loss does not necessarily outweigh the utility of attempting to corrupt oracle data. The profit from corruption depends on where the data is being used, which could be within larger system than the blockchain itself [141]. Finally, a feeder might suffer *reputational loss* for providing incorrect data. One can imagine this would be the case if, for example, the Associated Press misreported the outcome of the 2020 US Presidential election after announcing that it would serve as an oracle for this event on Ethereum.

Another illustration of these options is Chainlink, which maintains a decentralized analogue to a leaderboard where feeders are ranked according to the amount of LINK (Chainlink's token) they stake, as well as their past behavior in providing data that is timely and found to be correct. Data feeders with the outlier data will be punished by losing their collateralized LINK tokens and reducing their reputation score on the reputation registry. The lost of tokens is a direct cost, while the loss of reputation could impact their future revenue.

5.4.7 Classification of Current Oracle Projects

In Table 5.3, we present a classification of several oracle implementations using the modular framework described in this section. This table showcases a wide variety of approaches, as well as some specialization on specific modules (*e.g.,* TownCrier and Deco on data source and Kleros on dispute resolution). We caution that blockchain projects can change how they work very quickly, new projects will emerge, and current projects will be abandoned. Table 5.3 has a limited shelf-life of usefulness, however the workflow itself (modules, submodules, and design choices) is based on general principles and intended to have long-lasting usefulness. We exclude modules described in section 5.5 from this table as the infrastructure and implementation can differ for different use-cases of the oracle. While we can find some sort of implementation for most of the projects listed in table 5.3, it is hard to determine which ones are deployed in "production". Many have testnet or sidechain deployments, some are academic work and some have code but not clear if there are actual users.

	Data Source	Selection Mechanism	Staking	Aggregation Mechanism	<u>P</u>rovider/<u>D</u>ata Vetting	Determining the Truth	Consequences (<u>S</u>lash, <u>B</u>an, <u>L</u>oss)
Oracle		Data Feeder			Dispute		
ChainLink [112]	API	Reputation, Staking	•	Statistical Measure	P	Statistical Measure	S
UMA [323]	Human, API	FCFS†	•	×	D	Staking	S
Augur [257]	Human	Single Source⋆	•	×	D	Voting	S
Uniswap [324]	Smart Contract	×	×	TWAP	×	×	×
MakerDAO V1 [207]	Human, API	Centralized Allowlist	×	Median	×	×	×
MakerDAO V2 [207]	Human, API	Decentralized Allowlist	×	Median	P	Voting	B
NEST [239]	Human	×	•	×⋆⋆	D	Arbitrage	L
Band protocol [266]	API	Random Selection	•	Statistical Measure	P	Staking	S
Tellor [85]	Human, API	PoW	•	Median	P	Staking	S B
ASTRAEA [7] TruthCoin [302]	Human	Staking	•	Mode	D	Voting	S
Provable [31] PriceGeth [120]	API	×	×	×	×	×	×
DIA Oracle [105]	API, Smart Contract	×	×	×	D	Staking	B
DECO [357] TownCrier [356]	HTTPS	×	×	×	×	×	×
API3 [30] \w Kleros [194]	Oracles	Decentralized Allowlist	•	Statistical Measure	P	Voting	S B

Table 5.3: A classification of the existing oracle implementations using the modular framework described in Section 5.4.
• indicates the properties (columns) are implemented in the corresponding oracle (rows), and × indicates the property is not applicable.
† First Come First Serve ⋆*The Market Creator assigns the designated reporter* ⋆⋆ *The series of reported prices will be sent to requester without aggregation (See 5.4.6.1)*

5.5 Interacting with the Blockchain

While the initial inputs to an oracle are generally *off-chain* (except for pulling data from another smart contract) and the final output is by definition *on-chain*, the oracle designer will choose to implement the intermediary modules—data feeder selection, aggregation and dispute resolution—as either off-chain or on-chain. Generally, on-chain modules are preferred for transparency and immutability, while off-chain modules are preferred for lower costs and greater scalability.

To illustrate, Chainlink and NEST Protocols were ranked #5 and #7 respectively in gas usage amongst all DApps on Ethereum.[4] This ranking was achieved mainly because they implement all modules fully on-chain. Later, Chainlink implemented an off-chain reporting (OCR) protocol [46] with the goal of reducing the gas costs associated with on-chain transactions. This protocol uses digital signatures to authenticate feeders and a standard (*e.g.,* Byzantine fault-tolerant [60]) consensus protocol between Chainlink nodes.

At some point, an oracle system must move on-chain and start interacting with the underlying blockchain. We assume for the purpose of illustration that Ethereum is the blockchain being used. Data flow from an off-chain module to a smart contract involves the following three components which we detail in this section.

5.5.1 Off-chain Infrastructure: Assuming at least one module is off-chain, an infrastructure is required to monitor requests for oracle data from the blockchain, gather the data from the data sources, implement a communication network between data feeders, and create a final transaction to be sent to the blockchain infrastructure.

5.5.2 Blockchain Infrastructure: Off-chain infrastructure will pass the data as a

[4] Based on Huobi DeFiLabs Insight on September 2020 [168]

transaction to blockchain nodes, which relay transactions and use a consensus algorithm agree on new blocks. The nodes run by miners are discussed in particular as they dictate the order of transactions in every block they mine.

5.5.3 Smart Contracts: The transaction triggers a state change in a smart contract on the blockchain, typically a contract owned by the oracle which is accessible from all other contracts. Alternatively, the oracle could write directly into a data consumer's contract (called a *callback*).

5.5.1 Off-chain Infrastructure

Depending on the oracle design, there can be different types of off-chain infrastructure. If financial data is pulled from Uniswap's oracle [324], there is no off-chain infrastructure needed because the oracle is already a fully on-chain oracle. For other applications, off-chain infrastructure could consist of a single server (*e.g.,* TownCrier [356]) or many nodes that intercommunicate through their own consensus protocol (*e.g.,* Chainlink OCR [46]). Availability and DOS-resistance [295] are core requirements of off-chain infrastructure, specially in oracle systems working with time-sensitive data and high update frequency. In this section we describe different possible components of the off-chain infrastructure.

5.5.1.1 Monitoring the Blockchain

For oracles that are capable of returning a custom data request made on-chain (called *request-response oracles*), every data feeder needs to monitor the oracle's smart contract for data requests. The common implementation consists of a server subscribing to a blockchain node for specific events.

5.5.1.2 Connection to Data Source

The data feeder requires a connection to the data source 5.4.2 to fetch the desired data. This connection can be an entry point for an adversary to manipulate the data however it is possible to mitigate this issue by integrating message authentication (recall source authentication in Section 5.4.3.1). Examples include relaying HTTPS data (*e.g.,* Provable [31] via TLSNotary [320]) or from trusted hardware enclaves (*e.g.,* TownCrier [356] via Intel SGX [86]). Vulnerabilities with the web-server or SGX itself [44] are still possible attack vectors.

5.5.1.3 Data Feeders Network

In order to increase the scalability of the oracle network, multiple data feeders might aggregate their data off-chain (*e.g.,* Chainlink OCR [46]). In OCR, a leader is chosen from the participants to gather signed data points from other nodes. Once consensus is achieved on the aggregated set of data, the finalized data, accompanied by the signatures, is transmitted to the blockchain node. This reduces the costs as only one transaction is sent to the blockchain, while maintaining similar security as having each chainlink nodes send the data themselves.

Like any network system, availability is essential to the operation of the oracle. To illustrate, in December 2020, MakerDAO's oracle V2 had an outage due to a bug in their peer-to-peer data feeder network stack [143]. We do not summarize all the literature on peer-to-peer network attacks, but denial-of-service attacks [346] and sybil-attacks [109] are critical to mitigate to ensure the availability of the network and the oracle.

5.5.1.4 Transaction Creation

In order to submit data to a blockchain, the data feeder is required to construct a valid blockchain transaction that includes the requested data. This transaction must be signed with the data feeder's private key to be validated and authenticated on-chain. The data feeders must protect the signing keys from theft and loss [118], as this key can be used to impersonate the oracle.

Transactions compete for inclusion in the next block by offering different levels of transaction fees, known as the gas fee in Ethereum. In time-sensitive oracle applications, the relay must specify an appropriate amount of gas according to market conditions. For instance, on 'Black Thursday' in March 2020 [343], the Ethereum network was congested by high fee transactions and some oracles failed to adjust their price feed. To mitigate this problem, the module which is responsible for creating the final transaction must have a *dynamic gas* mechanism for situations where gas prices are rapidly climbing. In this case, pending transactions must be canceled, and new ones must be generated with higher gas price, which may take a few iterations to get in. Dynamic fees depend directly on the network state and require a connection to the blockchain node to estimate the adequate gas price.

In addition, the data feeder's sending address on the blockchain must have sufficient funds to be able to pay the estimated gas price. It is crucial for the availability of the oracle that the data feeders monitor their account balance as spam attacks might drain their reserves with high gas fees, as happened to nine Chainlink operators in September 2020 [90].

5.5.2 Blockchain Infrastructure

In this section, we discuss the blockchain infrastructure that is required by any entity interacting with the blockchain. While this infrastructure is not specific to oracles, we illustrate key points that can impact oracle availability.

5.5.2.1 Blockchain Node

A blockchain node relays transactions to the other nodes in the network for inclusion in the blockchain. The node is responsible for storing, verifying, and syncing blockchain data. The availability of nodes is very important for the oracle system, as a blocked node cannot send transactions. Extensive research on network partitioning attacks apply to decentralized networks, with the main objective of surrounding an honest node with the malicious nodes [328, 240, 359, 156, 160]. This results in the node believing it is connected to the blockchain network when it is not.

5.5.2.2 Block Creation

Transactions that have been circulated to the blockchain network are stored in each node's `mempool`. Mining nodes select transactions from their `mempool` according to their priorities (*e.g.,* by highest gas price as in Geth [124], while respecting nonces). Front-running attacks [122, 97] try to manipulate how miners sequence transactions. For example, someone might observe an unconfirmed oracle transaction in the mempool, craft a transaction that profits from knowing what the oracle data will be, and attempt to have this transaction confirmed before the oracle transaction itself (called an *insertion attack* [122]). This might be conducted by the miner themselves. In this case, it is called *transaction reordering*, and the profit miners stand to make from doing this is termed *Miner Extractable Value (MEV)* [97]. Other nodes or users on the network who can act quickly and offer high fees can also conduct front-running attacks. Users might also attempt a *bulk displacement attack* [122] that fills the consecutive blocks completely to delay reported data from oracles. There could be a profit motive for this attack if the oracle data becomes expired, or if the data feeder's collateral is slashed and redistributed to the attacker.

Research on MEV (*e.g.,* Flashbots [140]) has shown the possibility of new type of attacks based on reordering the transactions, such that if there's a high profit for changing the order of some transactions in a (few) blocks, miner is incentivized to use his hash rate to perform a reorganization attack[5] [198], and profit from the execution of the newly ordered transactions. For instance, Uniswap uses the last price in a block

[5] Also referred to as *Time-bandit attacks* [97]

to determine the average price (TWAP), in which a miner can add new trades while reordering the past trades with the goal of manipulating the price average to profit on other applications that uses Uniswap as price oracle.

5.5.2.3 Consensus

The goal of the consensus algorithm used in the blockchain is to verify and append the next block of transactions to the blockchain. If the nodes do not come to agreement on a state change, a fork in the network happens with different nodes trying to finalize different forks of the blockchain. Given the network is decentralized, short-lived forks happens frequently in the network that generally are resolved within a few blocks [241]. All valid transactions in the abandoned fork will eventually be mined in the main chain, likely in a new order (called *reorganization* or a *reorg*).

A reorg opens the possibility of attacks by using known, unconfirmed, transactions from the abandoned fork. To illustrate, consider Etheroll [131], an on-chain gambling game where users bet by sending a number that payouts if it is smaller than a random number determined by an oracle. To prevent front-running from the mempool, the Etheroll oracle would only respond when a bet was in a block. Despite this mitigation, in April 2020, the Etheroll team detected an ongoing front-running attack on their platform [116]. The attacker was betting rigorously and waiting for small forks—collected by Ethereum in *uncle blocks*—where the original bet and oracle's random number response were temporarily discarded by the reorg. The attacker would place a winning bet with a high fee to front-run the original bet and eventual inclusion of the oracle's transaction in the reorganized chain. A general principle of this attack is that even if oracle data bypasses the mempool and is incorporated directly by miners, front-running through reorgs is still possible.

There are two solutions to front-running through reorgs. The first is to delay the settlement of the bet by a few blocks to prevent issues caused by small reorganization forks. The second is to incorporate a hash of the request (*e.g., request-id*) in the response to prevents the request (*e.g.,* bet) from being swapped out once the response (*e.g.,* random number) is known.

Other consensus attacks [150, 34, 160] exist but are less related to oracles. We omit discussion of them.

5.5.3 Smart Contracts

Although oracles are usually designed to be the source of truth for on-chain smart contracts, some smart contracts can also be used as oracles by others even though they were not designed with the oracle use-case in mind. To expand this idea, oracles could be a *'an end in itself'*, which is to say they are designed specifically to be used as a source of truth. These oracles fetch the data from external sources(5.4.2) and make it available on-chain (*e.g.,* PriceGeth [120]).

By contrast, *a means to an end* oracle is a contract that produces useful data as a byproduct of what it is otherwise doing. Examples are on-chain markets and exchanges like Uniswap and other automated market makers (AMMs). The markets are designed for facilitating trades but provide pricing information (*price discovery*) that can be used by other contracts (*e.g.,* margin trading platforms) as their source of truth.

In this section we dive deeper in the relationship between the oracle's smart contract and the data consumer smart contract. We start by defining possible interaction models, and then discuss specific issues related to the oracle's contract and the consumer's contract.

5.5.3.1 Oracle Interaction Models

A distinction in the oracle design is whether the interaction between with the consumer's contract is implemented as a *feed*, a *request-response*, or the related *subscribe-response.*

A *Feed* is a smart contract system that publishes the data for others to use. It does not require any requests to fetch the data and using an interval to update the data on its smart contract (*e.g.,* Maker DAO Oracle [207]). From a technical aspect, in order to use a feed oracle, the data consumer smart contract only needs to query the oracle's smart contract and no additional transactions are needed.

The *Request-Response* model is similar to a client-server API request on traditional web development. The requester must send a request to the oracle's smart contract, which then is picked up by the off-chain module of the oracle to fetch the requested data from the data source. The data is then encapsulated in a transaction and sent back to the data requester smart contract through the oracle's smart contract. Due to the nature of this design, at least two transactions are needed to complete the work flow, one from the requester and another for the responder.

The *Subscribe-Response* model is similar to Request-Response with one main difference, the request does not need to be in a transaction. If there is pre-arranged agreement, the oracle will watch for emitted events from the requester smart contract and respond to the requests. Alternatively, the requester is allowed to read the feed through an off-chain agreement (*e.g.,* API3 [30]).

5.5.3.2 Oracle's Smart Contract

In the oracle designs that implement some modules on-chain, the oracle's smart contract could include data feeder selection (Section 5.4.4), aggregation (Section 5.4.5), and dispute resolution (Section 5.4.6). Additionally, the oracle's smart contract can be used as the data feed storage for other smart contracts to read from, or to authenticate the oracle's response on the consumer smart contract. In the *feed* model, the oracle's smart contract is where the consumer fetches the oracle data from. In the *Request-Response* model, the data consumer smart contract (defined below in Section 5.5.3.3) requires knowledge of the oracle's smart contract's address in advance, for the initial request and also verification of the oracle's response. For the rest of this section, we discuss potential attacks on the oracle's smart contract.

Implementation Flaws There are many known smart contract vulnerabilities that have been extensively discussed [82, 66] and possibly could affect the legitimacy of the oracle system.

In many DeFi projects, a common design pattern is to use on-chain markets, such as Uniswap, for the price oracle, however, these systems were not designed to be used as oracles and are prone to market manipulation. The end result is that currently, the most prevalent attack vector in DeFi is oracle manipulation [91]. To illustrate this attack, consider the lending (and margin trading) platform bZx. It fetched prices from KyberSwap, a decentralized exchange, to calculate the amount of collateral of one cryptoasset is needed to back the loan of a different asset. In one attack on bZx [255], the attacker used a *flash loan* to manipulate KyberSwap's sUSD/ETH exchange rate. The attacker then borrowed ETH with insufficient collateral because the bZx contract believed the collateralized sUSD was worth much more than it actually was. When the attacker absconded with the borrowed ETH, forgoing its collateral, and then unwound its other positions and repaid the flash loan, it profited at bZx's expense. Arguably bZx (the data consumer) is the flawed contract but the ease in which KyberSwap (the oracle contract) could be manipulated was not well understood at the time either. In reaction, decentralized exchanges embraced their role as a price oracle and hardened themselves against price manipulation by using aggregation methods like the *Time-Weighted Average Price* (TWAP) (described in Section 5.4.5).

Governance In order to remove the centralization of control in many DeFi projects, a governance model is introduced that uses a native token for voting and staking. The governance model for an oracle could propose, vote, and finalize changes to system variables like the approved data feeders on the oracle's allowlist or various fees.

While a decentralized governance model removes the trust in a central entity, it does not remove the possibility of a wealthy entity (a *whale*) taking control of the system by accumulating (or borrowing [268]) enough tokens to pass their proposals. In addition, logical issues in the governance implementation could result in tricking the voters into approving a proposal that has malicious consequences [233].

As an example, in the MakerDAO platform, MKR token holders can vote to change parameters related to Maker's oracle module [207]. An attacker in October 2020, used a flash loan to borrow enough MKR tokens to pass a governance proposal, aimed to change the list of consumer smart contracts and obtain read access to the Maker's oracle [204]. It could be more dangerous if the attacker planed to change the other parameters of the oracle such as *Whitelisted data feeders* or *bar* parameter : the sufficient number of data feeders for data feeder selection module. Potentially an attacker may pay a bribe to the MKR holders to buy their votes, or use a *Decentralized Autonomous Organization (DAO)* to pay for the votes without having ownership of the tokens [99].

5.5.3.3 Data Consumer Smart Contract

The final point in the oracle workflow is the smart contract that needs the data for its business logic. Aside from any possible code vulnerabilities in this smart contract, there are common implementation patterns concerning the oracle workflow.

In the *feed* model, the data consumer smart contract relies on oracles to fetch the required data in order to function as intended. It is essential to use oracles with multiple data feeders and a proper aggregation methods. To illustrate the importance, consider the lending service Compound [81] which initially only used Coinbase Pro as their data feeder without any aggregation mechanisms [101]. In November 2020, a faulty price feed on Coinbase Pro, resulted in undercollarization of Compound loans and a liquidation of $89 million dollars of the collateral. This could have been prevented by using an oracle with sufficient data feeders and a proper aggregation mechanism.

Due to the commonality of this issue, there has been some Ethereum Improvement Proposals (EIPs) to standardize the interface of the oracles implementing a *feed* (*e.g.,* EIP-2362 [310]). An interface would allow data consumer smart contracts to easily switch between feeds or use multiple oracle feeds in their logic.

In the *request-response* model, the data consumer smart contract sends a request for specific data to the oracle's smart contract. In some projects this request contains more information like the data feeder selection method, aggregation algorithm and parameters for dispute phase (*e.g.,* Service Level Agreement in Chainlink). It is crucial that the data consumer smart contract, verifies the authenticity of the oracle response. Failure to verify the oracle's response could result in malicious data injection in the data consumer smart contract. To illustrate, the insurance service Nexus Mutual [233] implemented an oracle's response function (or callback) without any proper access control. This opened the possibility of unauthorized entities providing data updates which would be wrongfully assumed to have originated from the oracle's smart contract.

5.6 Concluding Remarks

In this chapter, we described a specialized modular framework to analyze oracles. After our systematization, we present the following discussion points and lessons learned from our work.

1. Many oracles projects introduce their own governance tokens that are used to secure the oracle system (*e.g.,* through staking). Two conditions seem necessary: the market capitalization of the token stays material and the token is evenly distributed. More consideration should be given to leveraging an existing token with these properties (even a non-oracle token) instead of creating new specialized tokens [54]. Also, a collapse in the value of the governance token threatens the entire system.

2. Oracle systems with on-chain modules are expensive to run on public blockchains like Ethereum, which prices out certain use-cases that consume a lot of oracle data but do not generate proportional amount of revenue (*e.g.,* Weather data).

3. Diversity in software promotes resilience in the system. If the oracle market coalesces behind a single project, a failure within this project could cause cascading failures across DeFi and other blockchain applications.

4. While determining the profit from corrupting the oracle is a promising approach to thwarting manipulation (by ensuring the cost of corruption is greater), one can never capture the full extent of the potential profit. Attackers can profit

outside of Ethereum by attacking oracles on Ethereum [141].

In summary of this chapter, the framework we present facilitates a modular approach in evaluating the security of any oracle design and its associated components that exist today or to be implemented in the future. As an example, the level of centralization can be measured using choke points such as aggregation 5.4.5, or how the data is proceeded to the blockchain 5.5.1. In order to design a secure oracle, all modules must be rigorously stress tested to make sure it cannot be gamed by participants or malicious actors. In addition, many security auditors and analysis tools are specialized in detecting oracle-related attacks through code review of the smart contracts. Specially with the rise of DeFi smart contracts, the importance of a secure oracle system remain a paramount component of the decentralized blockchain ecosystem.

Chapter 6

Blockchain Audits: from Existence to Internal Controls

This chapter is based on the paper "Systemizing the Challenges of Auditing Blockchain-Based Assets" [262] published in the American Accounting Association Journal of Information Systems (JIS 2021, Volume 35, Issue 2). This paper was co-authored with Erica Pimentel, and supervised by Jeremy Clark and Emilio Boulianne. This chapter is partially rewritten to emphasize my individual contributions to the subject.

6.1 Introduction

In the previous chapters, we looked at several subtle technical issues with how a blockchain works and its limitations. The main conclusion is that if mis-understood or not mitigated, such issues can result in unforeseen consequences. These consequences can bring harm to the company behind the technology, the open source community developing it, and the users. Protecting users is important to government regulators, and protecting businesses is important to both regulators and to investors with a financial stake in the business.

One tool used by investors and regulators to protect businesses is to oblige or mandate it to undergo an audit of its financial position, as conducted by a highly regulated and arms-length audit firm. Another tool that is typically not legally mandated, but is often demanded by users and investors, is a technical audit of all protocols and software used in the solution. These two types of audits are not mutually exclusive. Financial auditors may take into consideration the results of technical audits when assessing the risks of a company holding blockchain-based assets or deriving profits from blockchain-based services.

The purpose of this chapter is to showcase one way the (occasionally theoretical or hypothetical) examples given in the previous chapters actually matter in the real world. Throughout this chapter, we focus on the financial audit of the crypto-assets on a company's balance sheet, as opposed to the technical audit (*e.g.,* security code review) of the smart contracts servicing these assets. However, we will consider both and conclude that they overlap—the technical audit of the smart contracts can be used as a tool to assist the financial auditors in their process (*e.g.,* Section 6.4.2).

Our research has uncovered that major accounting firms are hesitating to provide certification in the blockchain sector due to a perception of insurmountable business risk associated with these clients. Auditors believe that due to a complex and rapidly changing technological environment, they have yet to develop the in-depth knowledge of their clients' blockchain businesses in order to perform an audit. Due to the lack of guidance, standards and regulations in this space, auditors are reticent to take on new clients in a sector that has been subject to numerous frauds [1]. Accordingly, the inability to obtain audited financial statements presents a key barrier to investment and growth in the blockchain sector.

Put plainly, should the subtle risks we repeatedly illustrate in Chapters 3–5 frighten an auditor, who generally will not have deep technical knowledge, and cause them to avoid any company with crypto-assets on their balance sheet? This chapter aims to bring forth issues associated with auditing blockchain-based assets by making parallels to traditional financial statement audits. We cannot answer this conclusively, however our research points to several paths forward, which we compare.

6.2 Background and Motivation

6.2.1 Financial Audits

In certain common circumstances, firms operating in the blockchain space will require their financial statements to be audited by an external firm. Annual audits are legally mandatory for publicly traded companies in most countries, and audits might also be required when a firm borrows from a bank or raises capital from investors. Auditing is performed by a third party, who is independent of the firm being audited. The auditor's role is to provide an opinion on the financial statements of the firm, and to provide reasonable assurance that the financial statements are free of material misstatements or undeclared risks. The auditor's opinion is based on the audit evidence obtained during the audit process.

The blockchain industry is composed of firms (raising more than $15B in unregulated funding and $2.5B in venture capital in 2018 [75]) that issue and manage cryptoassets (worth a combined $1.64T at the time of writing [77]). While this market is experiencing rapid growth, it is dominated by startups that lack the financial sophistication and maturity of similarly valued traditional firms, and that rely on outside funding to develop and grow. These small firms will require audited financial statements to obtain traditional forms of credit such as bank loans or to gain access to public markets. For instance, SEC registrants must file audited financial statements that have an unqualified audit opinion (except under limited circumstances) [289]. If registrants are unable to provide unqualified audited statements, they will be unable to raise capital on public markets.

At the time of writing, auditing firms are hesitant to provide audit opinions to the blockchain sector. Furthermore, several crypto companies such as Impak Finance (who undertook the first legal ICO in Canada [103]), Hut 8 Mining Corp., Vogogo Inc. and DMG Blockchain Solutions Inc., were placed on cease trade for failure to produce timely financial statements when their auditors abruptly stepped down and the companies were unable to find replacement auditors [265]. Therefore, the inability to obtain audited financial statements is a pressing issue as both new and existing firms are having difficulty finding auditing firms who wish to provide opinions to crypto companies.

6.2.2 Cryptoassets

We use the term cryptoasset and cryptoliability to refer to listings on a firm's balance sheet that exist and are transacted using blockchain technology and have some tradable value. This includes cryptocurrencies (e.g., bitcoin and ether) as well as tokens (*e.g.,* ERC20) issued by smart contracts running on a blockchain. Among others, the main categories of tokens are: (1) *Access tokens*: a service is developed that requires its own custom tokens for using the service; (2) *Backed tokens*: a token issuer claims to be holding something valuable (material or digital) in reserve, and the token represents a claim on these reserves; (3) *Equity tokens*: a firm issues tokens to represent ownership shares of the company; and (4) *Collectable tokens*: the token itself is offered as a contemporary collector's item.

6.2.3 Ethereum Address Types

In Ethereum, there are two types of addresses: externally owned accounts (EOA) and smart contracts. An EOA is the equivalent of an address in Bitcoin. The address value itself is based on a hash of the user's public key. The address can be used to send a transaction, which is signed by the EOA's associated private key.

Recall that Ethereum allows the creation of smart contracts. A smart contract is generally created by an EAO, as a special kind of transaction different from sending

ETH (contracts can also create other contracts, which we generally do not consider here). During contract creation time, Ethereum designates an address for the contract. This value is derived pseudorandomly from details about the contract itself and thus does not have a corresponding private signing key. Thus anything done with the contract's address (such as sending ETH, tokens, calling functions on other contracts, *etc.*) must be coded into the functionality of the contract itself. At contract creation time, Ethereum copies the code of the contract into storage. The code is either provided directly or indirectly (through instructions that Ethereum can execute for obtaining a copy of the code). The code is stored at the contract address and the code cannot be changed with only one exception: executing the `selfdestruct` opcode.

ETH and other assets sent the address of a smart contract cannot be transferred out of the contract without support from the code of the contract. Contracts typically implement access control, the simplest being the designation of a single EOA that can perform sensitive tasks like transacting the assets belonging to the contract's address.

6.2.4 Multisig

A more involved access control pattern is a multi-signature (multisig) where more than one EOA (or contract address) must support the operation. For example, a multisig might require support from 7 of 12 addresses. There are different possible implementations and configurations on how each multisig smart contract can be set up [135].

An alternative to multisig is a threshold signature, where a single EOA is specified in the contract but off-chain, the signature has been set up such that 12 key-shares exist and any 7 of them is required to form a signature value [361]. fix this citation to something standard, like Goldfeder and Gennaro for threshold ECDSA. Thus, seeing a contract with a single EOA suggests but does not conclusively demonstrate single party control. Threshold signatures have one logistical drawback: the algorithm for creating the key shares generates a random share (so that it has the correct mathematical structure). Say Alice is participating in two different access control groups, only with multisig can she use the same EOA in both groups. Next assume Alice is a smart contract address (for example, wallet software like Gnosis Safe). Since contracts are assigned address values by Ethereum and have no corresponding public key, they cannot be used with a threshold signature scheme. They also cannot be directly used with multisig, in the sense that the contract cannot sign a message, however the multisig contract can be programmed to accept a message transferred from the contract as equivalent to a signature. Finally if Alice wants to change the EOA used in an access control group, with multisig, this change does not impact other members of the group. These are practical reasons why multisig is preferred over threshold signatures in industry.

6.2.5 Related Work from Auditing

In Chapter 2, we reviewed the academic literature and the technical details of blockchain technology. Here we briefly survey the accounting and auditing literature as it pertains to the technology. Blockchain technology has been referred to as an irrefutable record-keeping system [89], however, it does not verify whether a transaction was correctly accounted for under financial reporting rules or has a legitimate business purpose. While a blockchain verifies whether a transaction occurred and at what amount and on what date, it does not provide an examination of the internal controls that underlie the financial reporting process to prevent or detect fraud and errors, as would a financial statement audit. An audit is much more than a verification of routine transactions, being a holistic assessment of the robustness of a firm's internal controls, financial reporting policies and the reasonability of significant estimates. This activity cannot, in our view, be supplanted by just the blockchain.

A challenge for any auditor taking on a mandate in this space is demonstrating that they have sufficient competence to address the relevant technological risks. One way to achieve this would be for auditors to obtain the necessary level of technical knowledge through training [93]. Another way would be to engage technical experts to join the audit team. Bauer *et al.* [24] explore the relationship between IT specialists and audit teams to find that the quality of the relationship between these parties can impact the quality of the audit evidence gathered. Hirsch [163] studies the spatial distance and domain knowledge distinctiveness between auditors and IT specialists to suggest that auditors will rely differently on IT specialists whether they possess similar or different knowledge levels, and whether or not the specialists are in-house. In our research approach, we combine a multi-disciplinary group of blockchain experts and auditors, in addition to our own professional experience, to demonstrate that it is possible for auditors and technology professionals to work together to combine their mutual knowledge.

Broby *et al.* [50] provide a broad outline of issues that the auditor faces, such as forks, transaction malleability, and third-party custody, among others, coming to the conclusion that audits are not sufficiently robust in their current format to tackle the challenges of blockchain. We, however, disagree and will demonstrate how many of the issues with auditing the blockchain are surmountable, subject to certain caveats.

Some have explored how to exploit the blockchain's decentralized ledger technology for accounting purposes. Grigg [151] describes a system he refers to as "triple entry accounting," whereby a digitally signed receipt is created to represent a form of mutual authorization for a transaction between a buyer and a seller that a transaction has been carried out according to their specifications. This digitally signed receipt, coupled with a double-entry accounting system, complete a "triple entry system," which can leverage the power of financial cryptography to provide confidence over each accounting transaction entered into a company's accounting records. A blockchain could be used to take on this intermediary role and act as a centralized, secure repository for accounting records. Coyne *et al.* [87] take issue with using a blockchain in accounting settings, namely due to challenges surrounding confidentiality and the ability for a consensus mechanism to accurately verify transactions. Rückeshäuser [279] argues that blockchain systems based on a proof-of-work model provide an opportunity for management override of internal controls as the firm will likely hold a majority of a private blockchain's computing power. Dai *et al.* [96] raise

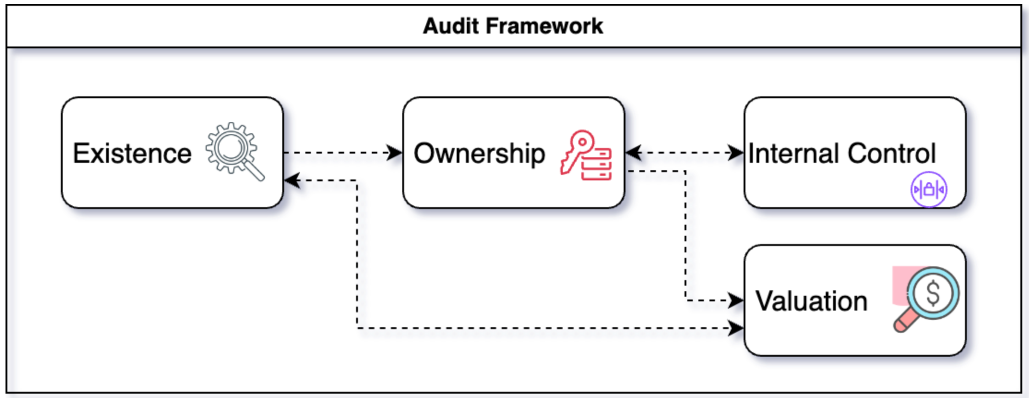

Figure 6.1: Audit Framework for Cryptoassets. Simply put, a cryptoasset *exists* when it has a material *value* and is *owned* by an entity. Ownership requires *internal control* to keep the asset accessible and safe.

the potential for the blockchain to support the audit function by enabling continuous assurance. For instance, the use of a blockchain as a data depository could allow information to be updated (and audited) in real time. Abreu *et al.* [1] and Smith [61] describe opportunities to replace traditional audit procedures such as bank confirmations with blockchain-enabled solutions. However, the deployment of these tools remains a challenge due to their lack of widespread acceptance.

6.3 Framework

We first set out to organize auditing issues into themes and establish a framework for analyzing them. In our paper [262], we report on an extensive set of interviews we conducted with auditors at large and medium sized firms. Applying thematic analysis to the interviews, we developed the framework in Figure 6.3. The framework breaks auditing issue into four themes: existence, ownership, valuation, and internal control.

This can generally be applied to any asset, however we consider how it applies to crytoassets specifically below.

6.3.1 Existence

Auditors need to establish that financial statements, as reported, are free of material misstatements. One area that was repeatedly cited as a challenging area for our respondents was the existence of cryptocurrencies. The major challenges to verifying existence are related to the reliability of a blockchain and the custodianship of assets. I am expecting subsections on these two challenges, but the second one is not there. Is it an existence issue in fact or internal controls?

6.3.1.1 Fraud Detection

Make up a token and say you have it. (Tulip Fund)

6.3.1.2 Reliability of a Blockchain

One challenge of auditing the existence of cryptoassets is simply due to their non-physical nature [262]. Unlike inventory or land that auditors can observe, auditors are required to find alternative evidence for these intangible assets. Evaluating the existence of a cryptoasset necessarily requires relying on a blockchain upon which a cryptoasset resides. Further, determining whether a given blockchain is reliable or not can prove difficult. Although blockchains are touted as immutable, not all blockchains are created equally. Hence, the ability to rely on a blockchain will depend on factors such as the robustness of the consensus mechanism, depth of the community supporting the blockchain, and reliability of the cryptography involved, among other things. If all nodes stop operating a blockchain protocol, the blockchain will effectively cease to exist. Consider Bitcoin: there is a large community supporting it, firms are involved, miners devote material hashpower to it, the cryptography and consesus mechanism have been reviewed by experts. The existence of BTC is less questionable than the native currency on an obscure altcoin.

Auditors must be able to evaluate the reliability of the blockchain they are relying on to provide evidence for the existence of cryptoassets. This is a challenge because it requires a level of technical expertise that many auditors do not possess. The challenge lies in determining how much work is involved in validating the blockchain itself. Whether a full code review or investigation of the underlying blockchain's cryptography is necessary will be a case-by-case decision. However, this may result in duplicate work if each firm is providing an in-depth review of each blockchain for each mandate. Best practice would involve firms developing a library of blockchains which have been tested (for instance, blockchains for which a code review has been done at a certain date) and which future engagement teams within the firm network can rely on to reduce the duplication of work.

Another challenge remains determining how to rely on transactions that are not on the blockchain. For instance, many exchanges pool or commingle the accounts of several clients. In a secondary ledger (not on the blockchain), they record the positions of each client and then may record offsetting positions of those clients in this secondary ledger and not actually record the exchange on the blockchain.

Auditors must be careful when evaluating audit evidence to verify the source for their existence support. While there are procedures that will allow the auditor to rely on the blockchain to validate existence, the auditor must validate that the transaction actually took place on that blockchain. If not, the immutability of the blockchain is irrelevant. I don't understand what this paragraph is saying, can you rework it?

Other existence points: I could make up a token and say it exists. Say I create a token contract but it does not follow ERC20, is it a token? How do you figure that out?

Shorten above

6.3.1.3 Forks and Airdrops

Existing accounting standards do not contemplate how to account for non-reciprocal transfers of assets like in the case of forks or airdrops.Justify this. Also we have standards for gifts, how is this different from airdrops? If a publicly traded company spins off a new entity, how is a stock spinoff different than a fork?

For example, in August 2017, when a hard fork created Bitcoin Cash from Bitcoin's main ledger, holders of Bitcoin before the fork were assigned Bitcoin Cash in the same amounts.

Expand a little, focus on existence, forward pointers to other aspects.

6.3.2 Ownership

Auditors must be satisfied that the assets reported on the company's balance sheet do in fact belong to the company, or if the client is operating as a custodian for third-parties (for instance, as an exchange who holds cryptocurrencies received from third-parties), that the assets held do in fact belong to the third party who claims to possess them. For traditional assets, firms might overcome the ownership issue in several ways. First, a client can demonstrate ownership of an asset with reference to a generally accepted official document. For instance, a property owner can demonstrate ownership of their building with reference to a deed. However, in a blockchain environment, no central authority exists to produce such official documents. Second, firms may engage a custodian to hold assets on their behalf, like a bank. This does not eliminate the issue of ownership but simply shifts the concern from the firm's audit to the audits of central custodians.

Therefore, addressing whether or not a client maintains ownership over their cryptoassets will depend on whether they hold their cryptoassets themselves (self-custody) or through a custodian.

6.3.2.1 Fraud Detection

Token exists but you don't actually own it.

6.3.2.2 Self-Custody of Assets

In the absence of legal registers to support ownership or documents bearing the name of the firm, the auditor must rely on the internal controls of the entity to obtain comfort over the ownership assumption [262]. A question emerges about what ownership means in this context both temporally and in terms of access to a private key. Temporally, auditors must distinguish between whether they are able to provide support for ownership over an asset at the date of conducting the audit procedure or at the end of the fiscal period under audit. For example, if an auditor verifies that a client owns bitcoin one month after year-end through procedures like signing messages or transferring small amounts to and from the key, this proves that the client controlled the asset on that date but says nothing about whether the client owned those assets at year-end. With inventory, in the absence of performing a count at year-end, the auditor may perform roll-back procedures to verify the transactions between the count date and the year-end date to obtain comfort over the year-end balance. To do so, this would require that the auditor ensure that ownership was maintained over the subsequent events period even if this is not what is required by the auditing standards. An auditor from our [paper] says:

> Do I need to prove that they retained ownership over the subsequent events period? The auditing standard doesn't say that I do. If the client loses ownership during the subsequent events period, do I need to unrecord the whole thing at year-end? Likely, this would be an issue of note disclosure.

> In the past, this has been used before for some fraud cases, but in those cases, it was because those transactions never really occurred.

What is this quote? Like from who? From Erica's interviews? Make it clear, it seems out of place

Determining when to test ownership to ensure that evidence is obtained at the correct date becomes a challenge. Best practice would indicate that, like inventory, auditors should test ownership as at the balance sheet date and may wish to provide note disclosure for significant events where ownership is lost.

A question emerges about what ownership really means in this context. A client may demonstrate that they have access to a private key, but this in and of itself does not demonstrate ownership. "With ownership, the risk of giving someone else access to the private key is no different than the corporate controller sharing his password to the company bank account with his spouse". Auditors must determine what types of procedures they can do to validate control and ownership in this context:

If a client uses self-custody, we can do different procedures to be able to approve ownership like small amount transfers or secret messages, depending on the protocol they're using.

These practices like small value transfers or the sending of secret messages are part of what are referred to as cryptographic proofs.

6.3.2.3 Cryptographic Keys

How is this subsection different from self-custody? A bunch of stuff repeats across both sections, can you merge?

For most cryptoassets, the asset is considered owned by Alice if Alice possesses a private signing key that can be used to digitally sign a transfer of the asset (See Section 2.1.1.1). Although alternative notions of ownership are possible to define, the idea of a signing key is foundational and seen with bitcoin, Ethereum (ETH), and ERC20 tokens. Thus, demonstrating knowledge of this key is necessary, but not sufficient, to demonstrating ownership. The most direct cryptographic technique is to use a so-called zero-knowledge proof of this private key, and to staple in some information identifying the context of the proof. For standard proofs, this is cryptographically equivalent to simply signing a challenge message with the key. Folklore protocols of sending small cash amounts from an allegedly owned account to the auditor to demonstrate control are also commonly noted in the literature. This offers similar security but may add ethical complexities for the auditor in accepting the amount transferred.

On reflection, I guess there is a difference between off-chain signing, OP_RETURN or equivalent, and sending small amounts. The second two leave on-chain evidence, while the first does not. So if I am sharing assets with another insolvent entity with different auditors, we are more likely to get caught by the second two methods than the first one.

We note that while cryptographic proof is necessary, it is not sufficient. A cryptographic proof simply demonstrates that the purported owner has access to the person holding the signing key. A malicious company might arrange for the owner of cryptoassets to engage in signing statements or moving test amounts fraudulently on their behalf. This issue is not new: an insolvent retail store might borrow inventory from elsewhere to inflate its assets during an inventory count. Auditors mitigate this by arranging a common date for all audits of physical inventory and, similarly, cryptographic audits could be synchronized on a fixed schedule to prevent the same assets from being counted for different companies in different audits [95].

One type of proof includes sending a small amount (the amount is random challenge) of money to the auditor from the private key, while another proof involves demonstrating that the client can respond to a cryptographically-protected message that only the private keyholder could open. Independently, neither of these procedures demonstrate that the client owns the private key. It should be noted that due to technical nuances of the cryptographic implementation, this requires tooling to be able to irrefutably verify the validity of the signature by the auditors [246]. However, auditors rely on the sum of several procedures to obtain reasonable assurance over this assertion:

> With ownership, it's not a specific procedure but rather the body of evidence that can be performed by signing messages, by testing internal controls, by understanding how the client protects passwords. The clear expectation of ownership is changing. In a traditional audit, the client represents that they own certain things. We see an invoice and we see

> that they own it. But did they really pay for it? Was it paid for by another company and consigned to them? We perform several procedures to feel comfortable enough to say that they have ownership at that point in time. Our expectations of what ownership means in this area is evolving.

Many clients are concerned about self-custody due to security risks over holding their own private keys and are turning to custodians to fulfill this function for them. Auditors must adapt their audit procedures to their clients' unique internal control environments and consider competing sources of evidence before coming to a conclusion about ownership.

You should also mention TEEs and encumbered keys — see paper on complete knowledge

6.3.2.4 Third-Party Custodianship

In a traditional audit context, the reliance on a third-party custodian is commonplace. Clients might have bank accounts with multiple banks or investment accounts with various brokerage houses. Auditors consider confirmations from these regulated custodians as high-quality evidence. Confirmations in the cryptoasset space are not as straightforward because the entities the auditor would be requesting confirmations from, such as a crypto-exchange, are not always regulated. This raises questions over the reliability of their responses.

In order to address the reliability of the confirmations from service providers such as cryptocurrency exchanges or other types of custodians, auditors look to the robustness of the internal controls at the service organization. The robustness of these controls is evidenced by the presence of a service organization control (SOC) report. In order to rely on the controls of a service organization such as a payroll provider or investment custodian, auditors often obtain SOC reports, which provide assurance over the processes and data security at the custodian. Two types of reports are available: an SOC 1 report provides assurance over the controls used by a service provider who processes financial data; an SOC 2 report provides assurance on controls over the processing of non-financial data in accordance with Trust Services Criteria [27].

While some blockchain firms have been able to obtain SOC reports, the mere fact of having the report is insufficient. Obtaining an SOC report is not a box-checking exercise and that the auditor must review the report carefully to understand which controls it has addressed and can be relied on.

6.3.2.5 Airdrops again

More commonly for ERC-20 tokens on Ethereum, the recipient of the tokens often does not have the option to reject the deposits, nor would they get notified of the new tokens received, which could result in issues with completeness. Can you add something here to make it more real? Like "author's personal account" received XX tokens over YY years. However, in some other instances, such as Optimism Airdrop [9], the recipient of the tokens must claim the tokens in order to receive them. In this case, the auditor must ensure that the client has claimed the tokens, and the price of the time of the claim, in order to include them in the financial statements.

6.3.3 Valuation of Cryptoassets

When values are reported on financial statements, they must be reported in the functional currency of the firm, meaning the primary governmental currency used. A challenge for blockchain entities is to determine the valuation of cryptoassets on the financial statement date or the conversion rate for sales and expenditures made throughout the year. Auditors must be satisfied that the values reported in the financial statements are accurate and represent the underlying economic reality [134]. This issue can be considered a subset of the "Oracle Problem" which is discussed further in the context of decentralized applications (DApps) and smart contracts in Chapter 5, however, the issue discussed here is more related to the valuation of cryptoassets on the financial statements of the company.

6.3.3.1 Fraud Detection

Token exits, you do own it, but it is worth less than you claim.

6.3.3.2 Fair Value of Cryptoassets

A significant obstacle for obtaining audited financial statements is the determination of a fair value for cryptoassets. Cryptoassets are often difficult to value because it is challenging to determine their underlying value and there may not be a generally accepted, quoted value to use as a reference. As an analogy, firms value foreign currencies at the closing rate on the transaction date, as reported by the central bank servicing the firm's area. No universal central bank offers rates for currency-like cryptoassets. At the time of writing, only a few Fortune 500 financial firms, *e.g.,* CME (Chicago Mercantile Exchange), offers a daily reference rate for bitcoin and ether, but not for most cryptocurrencies or cryptoassets. While bitcoin and ether enjoy around-the-clock trading across many markets, lesser-known coins, tokens, assets, or liabilities may trade slowly, and in low volumes. Generally speaking, low liquidity results in stale last sale prices and large bid-ask spreads. This is challenging but not unprecedented in financial auditing: privately held stocks and over-the-counter financial instruments share a similar profile. Auditors must familiarize themselves with the exchange markets for the cryptocurrencies held by their client to assist in validating their valuation.

6.3.3.3 Geographical Variation

The same cryptoasset might have different market values across different jurisdictions — because of market frictions, arbitrage does not resolve these differences [188]. If a firm applies International Financial Reporting Standards (IFRS), any financial assets measured at fair value that they hold must be determined with reference to their principal market (if available), which refers to the "market with the greatest volume and level of activity for the asset" [169]. Therefore, this standard precludes a firm from using the valuation of a cryptocurrency based on an obscure market price. Auditors will need to look carefully at the record of a client's trading activity to determine the location of a client's principal cryptoassets market and ensure that the assets are valued accordingly on the financial statements.

6.3.3.4 Fungability

Tainted ETH from Tornado Cash. Dusting attacks

6.3.3.5 Airdrops again

However, if existing accounting standards require the measurement of transactions at historical cost (what was paid for the assets), then recipients of Bitcoin Cash would report this new asset on their books at a value of $0. Certainly, this does not represent the true value acquired through the fork [340]. Therefore, auditors must address the issue of an accounting standard that does not contemplate how to measure the value of cryptoassets when considering whether the financial statements they are reporting on are accurate in all substantial respects. This sounds like a valuation issue, below, rather than an existence issue. What is the existence issue? If none, you can move it to valuation.

6.3.4 Internal Controls

Auditors must be satisfied that the financial statements they are reporting on are free of material misstatements. this sentence is a repeat In order to do so, auditors must obtain reasonable assurance that the internal controls of the client are effective. This is done by testing the design and operating effectiveness of the internal controls. In the context of cryptoassets, this is challenging because the adequate internal controls are novel to the auditors, often not well documented and are not always securely automated.

6.3.4.1 Segregation

Trade-off between ownership and internal controls. Safer internal controls like multisig make proviing ownership harder. Single EOA is good for proving ownership but made for internal controls.

Keys. System to protect keys. Policies around keys. Keys need to be used, policies around that, who allowed to do a transaction, how is logged. Cold wallets vs. hot wallets. Auditors role? Diversity/redundancy of software/hardware.

A related issue to the robustness of the internal controls relates to how the custodian segregates the assets in their possession:

> We also have heard that many of these custodians are co-mingling or combining assets into a single account or wallet. That muddies the water a bit and it's difficult in some type of SOC reports to understand what they are really doing to maintain a client's assets. There's a chance that there actually are no assets. If you've given your assets to someone else to hold for you, you may think that they still are yours and that they still exist, but that can be difficult to ascertain.

When evaluating whether they can rely on the representations from a custodian, the auditor should evaluate how the custodian segregates the assets in their possession.

In short, in order for auditors to validate ownership, their procedures will depend on whether the client has custody of their keys or uses a third-party. Self-custody will depend on the client's internal controls, while reliance on a custodian will depend on the ability to obtain comfort over the reliability of the custodian. Additionally, cryptographic proofs play an important role in the ability to rely on either party. In order to avoid double-counting of keys, an industry standard common date should be arranged to provide a generally agreed upon "state of the world" where keyholders can demonstrate ownership.

However, not all auditors may be as aware of the importance of SOC reports. One common method is to rely on confirmations from exchanges as a way to corroborate the existence and ownership of the client's assets, since third-party audit evidence is traditionally viewed as the highest quality of audit evidence for addressing these assertions.

This finding raises two issues. First, auditors believing they could rely on technologists at their firm to compensate for their lack of knowledge. However, having access to expert knowledge is not the same as deploying it. As we have cautioned throughout the paper, it is incumbent on auditors to develop a fundamental knowledge of blockchain technology to be able to leverage the skills of specialized professionals. And this synergy is only possible when auditors and technologists collaborate [24]. Second is the lack of guidance on how to audit cryptoassets. We believe that by providing rigorous sets of standards and through ongoing inspections by audit oversight bodies, a corpus of generally accepted auditing standards for this sector will develop.

These standards will provide guidelines that auditors who are not blockchain experts can use when attempting to make inroads into this sector.

6.3.4.2 Old Text

It is necessary for a firm to be able to demonstrate the existence and ownership of its crypto-assets, however it must further demonstrate that it has adequate procedures in place to prevent or detect fraud and theft. Consider a firm holding DAO governance tokens. Internally, it must ensure proper controls over the signing keys these DAO tokens are assigned to. External to the firm, the DAO smart contract maintaining these tokens on Ethereum's blockchain must itself be secure (which in the case of the DAO, it was not [291]). Auditing internal procedures over cryptographic keys is not unprecedented—maintaining certificate authority keys is critical to some security firm's financial prospects (a security breach at one firm, DigiNotar, bankrupted it [325]). However, the use of cryptographic keys is not yet commonplace in financial auditing.

You spend a while on multisig, is that relevant here? Can you motivate why it is not necessarioly sufficient with Ronan?

As discussed in Section 6.3.2.4, auditors rely heavily on SOC reports to obtain comfort over the internal controls of a custodian. However, SOC reports are not always available, and even when they are, they do not always cover the relevant controls for custody of cryptocurrencies. SOC reports require the company to provide a description of the system, including the controls in place, and the auditor to provide an opinion on the design and operating effectiveness of the controls. However, the SOC report is not a panacea. The auditor must still evaluate the report to determine whether the controls are adequate for the purposes of the audit. Additionally, the SOC report is only as good as the date it was issued. If the SOC report is dated after the balance sheet date, the auditor must obtain additional evidence to ensure that the controls were in place at the balance sheet date.

6.4 Case Studies

As is quite common in the business academic literature (compared to the computer science literature), we will illustrate a set of auditing issues through a set of case studies. For each case study, we present a realistic scenario. We then classify the auditing issues based on our framework. Finally we analyze the case for relevant examples where subtle technical nuances blockchain technology need to be understood, fitting with the theme of the dissertation.

While the case studies are fictitious (unless otherwise noted), they are rooted in real experiences in industry. My personal experiences include, but are not limited to, my role as: a blockchain engineer at the Bitcoin ATM company *Bitaccess*, a senior security auditor at *ConsenSys Diligence*, a chief technology officer (CTO) at the publicly traded company *Ether Capital* which runs Canada's first Ethereum ETF. Additional input came from my advisor Jeremy Clark, a research chair sponsored by *Raymond Chabot Grant Thornton*, and from my co-author Erica Pimentel who conducted interviews of auditors for our paper [262] (see the paper for the description of each interviewee). The case studies do not directly represent any of the companies mentioned above.

6.4.1 Case Study 1

Optimism is a software project for building a scalable 'layer 2' on top of Ethereum. At the time of writing, it represents the second largest layer 2 (next to Arbitrum).[1] The Optimism [249] team decided to airdrop `OP` tokens to the Ethereum addresses that met a criterion of activity on the Ethereum mainnet. In this section, we do not discuss what Optimism is, and how it works, and we only focus on the auditing challenges of the airdrop for the auditors and the cryptoasset holders (See Section 6.3.1.3).

The airdrop was done in two phases. The first phase was done in May 2022, and the second phase was done in February 2023. The first airdrop was done in a way that the recipient of the tokens must claim the tokens in order to receive them. This is different that many previous *push* airdrops, where the recipient of the tokens did not have the option to reject the deposits, nor would they necessarily get notified of the new tokens received (depending on whether their software client was active and configured to report new transactions).

Classification. This case study mainly is focused on the *existence* (See Section 6.3.1) of the tokens and how the auditor must develop a procedure to ensure that the tokens have been claimed, transferred, and accounted for in the financial

[1] `https://l2beat.com/scaling/summary`, accessed Mar 2024.

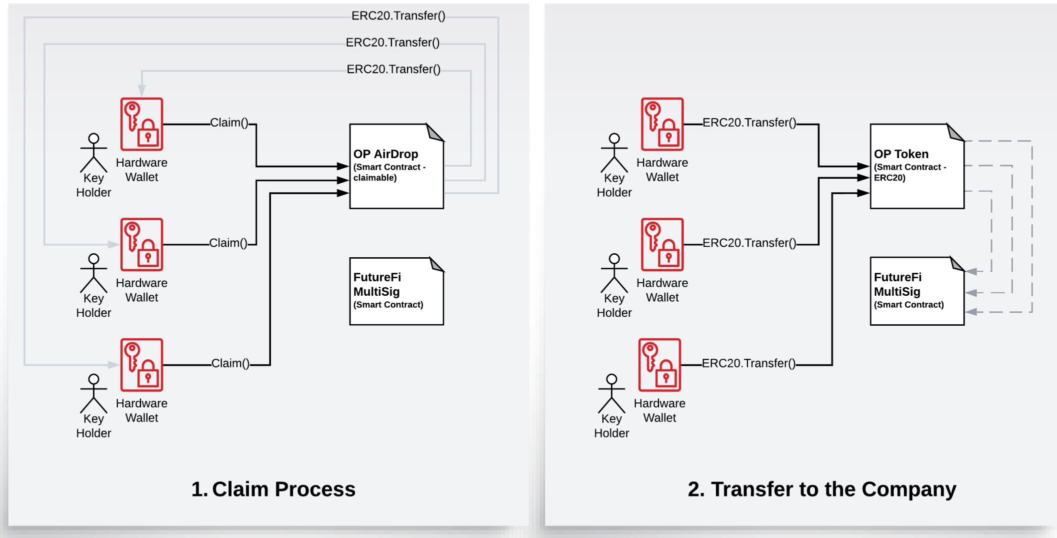

Figure 6.2: Case Study 1 - Optimism Airdrop First Claim Workflow for Multisig Custody

statements. However, such as the valuation of the tokens, and the internal controls of the client (in the claim process) are also important factors that must be considered by the auditor.

Analysis. There are some technical nuances that are worthy of mentioning. Assume a company has a multisig smart contract account as their main address, and 10 keyholders for the multisig smart contract (the threshold here does not matter). Each of the keyholders were included in the recipients list of the airdrop. In order for the company to receive the airdropped tokens, each of the keyholders must claim the tokens by sending a transaction (See Figure 6.2 for more details). For normal use of the multisig smart contract (*e.g.,* Gnosis Safe), the keyholders do not require an Ether balance to pay for gas, however this transaction requires an onchain transaction and gas. After the claim process is done, the tokens required to be transferred to the main address of the company, which required more gas to be paid. This process is cumbersome and requires manual review of the transactions to ensure the correctness of the process. To make the matter more complicated, the airdrop was done in two phases[2], and the second phase was done after the first phase was completed and only airdropped to addresses that have claimed and participated with the governance of the Optimism protocol [9]. The second phase of the airdrop did not need to be claimed and was sent to the eligible addresses, which made the auditing process more complicated and different from the first phase.

In here we assume all the technical details of the claim process has been handled by the technical team of the company, and the auditor only needs to ensure that the tokens have been claimed. It should be noted that this process by itself is not defined in the auditing standards or procedure and is a novel process that the auditor must develop a procedure for. In order to audit the existence of the airdropped tokens, the auditor must ensure that the client has claimed the tokens, the cost of the claim (gas), and the price of the time of the claim, in order to include them in the financial statements. This process is not straightforward and requires manual auditing of the client's Ethereum addresses. Additionally, the auditor must ensure that the client has claimed the tokens before the end of the fiscal year, otherwise the tokens must be excluded from the financial statements. There are many open challenges in this

[2]There were three consecutive airdrops, however, for the sake of simplicity in this case study, we will focus only on the first two, as the second and third airdrops were technically identical, rendering the examination of the third phase redundant.

process, as each transfer of tokens requires to be recorded and explained in the audit report. Additionally, many of these airdropped tokens, are not listed on any exchanges or have really low valuation to be considered material, hence the auditor must rely on the client's internal valuation of the tokens.

6.4.2 Case Study 2

Sean, the CTO of a publicly traded company, is preparing for the year-end audit of the company's financial statements. The company holds a significant amount of cryptoassets, and the auditor has requested proof of ownership of the cryptoassets. The company holds all the cryptoassets in a hardware wallet [118] stored in a safe deposit box at a bank. Sean, using the hardware wallet, with auditors as observers, creates a message based on the day's newspaper headline, and signs the message using the hardware wallet. The auditor, using the public key of the hardware wallet, verifies the signature and confirms the ownership of the cryptoassets.

In the following year, Sean upgrades the security of the company infrastructure by moving the cryptoassets to a multisignature smart contract. The keyholders are part of the management and another part board members, separated by responsibility and geographical locations for added security [57].

The following quarter, the auditors ask for proof of ownership of the cryptoassets and expect the similar procedure as the last year. However, the company has moved the cryptoassets to a multisignature smart contract, and the auditors must develop a new procedure to be able to verify the ownership of the cryptoassets, as the technical details of the custody has significantly changed.

> *One of our first considerations as it relates both existence and ownership is just to think through how the entity is maintaining custody of their assets. And in particular, what's the security mechanism around access to the private key? Which can vary from being held [in many ways] from online to some type of hot storage to on a piece of paper somewhere, to in a software tool that is not connected to the internet and kind of offline. Which of course, brings down the risk of it being hacked but then increases the risk of physical loss of the private key. We want to understand what our potential clients are doing related to that. Then, as many of them seem to do, we start to gather that they're using a third party around custody.* [262]

For the new year audit, almost none of the previous procedures can be reused, mainly due to the fact that previous year's audit, the premise of the ownership of the assets, where based on proof of the access to the private key (*e.g.,* signing a message, See Section 6.3.2). Nonetheless, the new year's audit, includes similar procedure for each keyholder, and a separate procedure for the multisig smart contract and the technical nuances.

Classification. This case study mainly is focused on the *proof of ownership* (See Section 6.3.2) of the cryptoassets and how the auditor must develop a unique procedure based on the technical details of how the company custody their cryptoassets.

Also this process is fairly technical and requires in depth knowledge of the cryptographic and blockchain tooling, and the smart contract implementations. There are also many factors regarding the internal controls of the client that must be considered by the auditor, such as how the keys (for each keyholder) are generated, how they are backed up and stored, and the multisig deployment.

Analysis. As described in Section 6.2, multisig smart contracts are commonly used by companies to securely store their cryptoassets. Unlike EOA addresses, smart contracts do not have a private key associated with them, hence the proof of ownership cannot be done using the digital signature verification. One approach is to fall back on the older method of transferring a small amount of the asset from and back to the smart contract address. Another is to prove the ownership by by verifying the code of the smart contract and the only verify the signatories of the multisig smart contract. This process is more complicated and requires manual auditing of the smart contract code and variables. Additionally, the smart contract can be upgraded or the signatories of the multisig smart contract can be changed at any time, hence the proof of ownership must be done at the time of every audit, and manually verified by technical experts.

6.4.3 Case Study 3

Bob is the CTO of a company holding cryptoassets named *FutureFi*. He believes the future of auditing is access to *Real-time Financial Reporting (RFR)*, and he

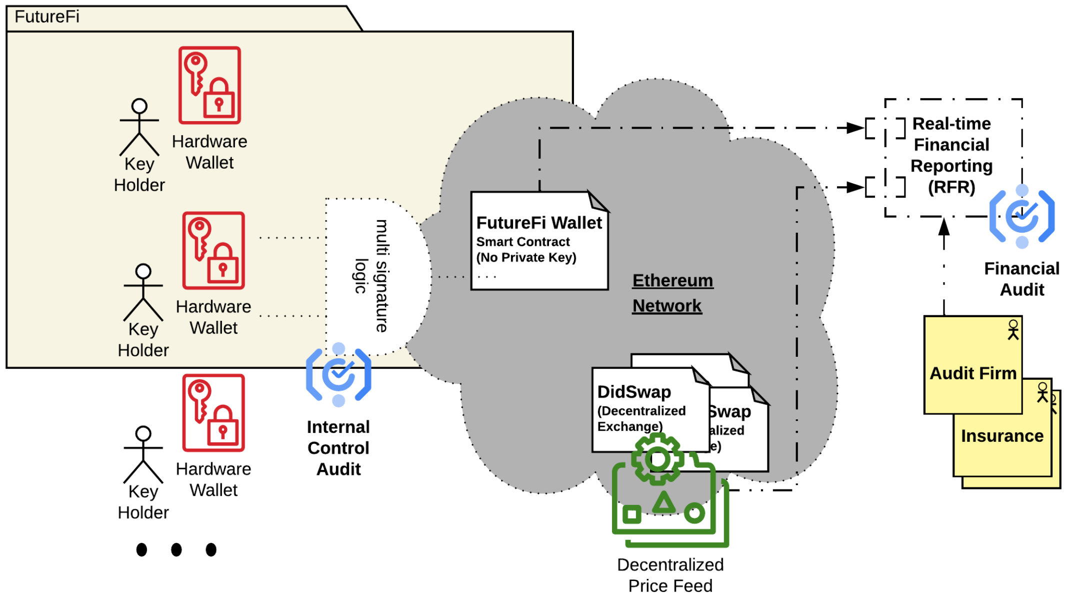

Figure 6.3: Case Study 3 - FutureFi Real-time Financial Reporting (RFR)

wants to be one of the first companies to implement real-time financial reporting and auditing (See Figure 6.3). Bob has developed a system that allows the auditors (or any other entity) to fetch the quantity of the cryptoassets held by the company from a blockchain full node and the valuation of the cryptoassets is done using a decentralized price feed as the oracle (See Chapter 5). This new system replaces the previous procedure of relying on the internal records of the company or use of third party block explorers. The previous methods of obtaining the quantity of the cryptoassets held by the company were prone to errors and software bugs that might have been challenging to identify without rigorous manual review of the code.

Bob's new system allows the auditors to fetch the quantity of the cryptoassets held by the company directly from the blockchain using their own (or trusted) node infrastructure with high degree of confidence. Additionally, the system allows the auditors to fetch the price of the cryptoassets from a decentralized price feed (Such as a Decentralized Exchange), which is mainly more transparent and in some extent more reliable than the centralized price feeds. Centralized price feeds are prone to manipulation and errors with no audit trail, possibly have downtime or delays, and are not transparent. The auditors are satisfied with the new system and the company is one of the first companies to implement real-time accounting and auditing.

Classification. This case study mainly is focused on the *valuation* (See Section 6.3.3) of the cryptoassets and how the auditor must develop a procedure to ensure that the price feed is reliable and not manipulated. Additionally, the auditor must validate the real-time financial reporting implementation for any accounting errors and review the internal control of how the quantity of cryptoassets and the price is fetched and included in the financial statements. We believe there are merits in real-time financial reporting standardization [21, 352] that can benefit both the auditors and the clients, however, the technical nuances of the implementation could result in unforeseen challenges for the accurate reporting of the financial statements. Dai *et al.* [96] raise the potential for the blockchain to support the audit function by enabling continuous assurance.

Analysis. In such scenario, the auditing process is more efficient and less prone to errors, as the auditors do not need to rely on the client's internal records for the quantity of the cryptoassets held by the company. However, this scenario is not without its own challenges. For example, the auditors must ensure that the price feed is reliable and not manipulated. Additionally, the auditors must ensure that the price feed is not delayed, as the price of the cryptoassets can fluctuate significantly in a short period of time. Any mistake in the price feed can result in significant changes in the audit snapshots, which can make the company insolvent for a short period of time. Any oracle manipulation attack, explained further in Chapter 5, can result in either over or under valuation of the cryptoassets, resulting in an inaccurate financial statement.

6.4.4 Case Study 4

Ian is an auditor working in one of the big 4 auditing firms. He is assigned to audit a company holding cryptoassets. This company uses a multisig smart contract to custody their cryptoassets. The multisig smart contract is a 2 of 3 multisig smart contract, and the signatories are the CEO, the CFO, and the CTO of the company. The company has acquired a SOC1 report from their custodian, and Ian is satisfied with the internal controls of the custodian. However, the company has a multisig smart contract, and Ian must ensure that the internal controls of the multisig smart contract are adequate. Ian is not familiar with the technical details of the multisig smart contract, and he must rely on the expertise of the company's technical team to ensure that the internal controls of the multisig smart contract are adequate.

To expand on the example, SOC1 report consist of two parts [202], the description of the system, and the auditor's opinion on the design and operating effectiveness of the controls. The description of the system includes the controls in place, and the auditor's opinion is based on the description of the system. However, the SOC1 report is not a panacea. The auditor must still evaluate the report to determine whether the controls are adequate for the purposes of the audit. In this scenario, Ian might consider asking for SOC2 report, as it might be more appropriate. SOC2 provides assurance over controls over the processing of non-financial data in accordance with Trust Services Criteria [27], which might be more relevant to the internal controls of the multisig smart contract, and mainly the procedure of how each new transaction is signed by the keyholders (*e.g.,* outgoing transaction authorization).

Classification. This case study mainly is focused on the *internal controls* (See Section 6.3.4) of the cryptoassets custody and key management. SOC reports that are commonly use can ensure the basics of the internal controls and security, however, technical nuances around key management and smart contract security are not covered by such reports. We believe there are merits in adopting standards, such as the C4 Cryptoasset Security Standard Audit [57], by the big 4 auditing firms, as it is a more appropriate standard for the internal controls of the cryptoassets. CCSS standard includes review of the private key generation process, secure backups, the multisig setup, and manual audit of the multisig implementation and source code.

Analysis. As described in Section 6.3.4, auditors must be satisfied that the internal controls of the client are effective. This is done by testing the design and operating effectiveness of the internal controls. In the context of cryptoassets, this is challenging because the adequate internal controls are novel to the auditors, and the expertise to evaluate and test these controls are not always available. Even if the company has acquired their SOC reports, it is still not sufficient to assume the adequacy of the internal controls. There are better suited standards that test the appropriate checks and balances for the internal controls of the cryptoassets, such as the C4 Cryptoasset Security Standard Audit [57]. However, these standards are not adopted by any of the big 4 auditing firms at the time of the writing.

6.5 Discussion

To enumerate the points discussed in section 6.4, we can see that the auditing process of the cryptoassets is not straightforward and requires technical expertise with a deep understanding of the auditing standards and procedures. This is a rare combination of skillets that is not common in the auditing industry. This is a challenge for the auditors, as they must develop a unique procedure for each client, and the lack of standardization can result in errors and mistakes in the auditing process.

Blockchain and smart contracts are labeled as *trustless*, however, the trust is in the code and the protocol to be implemented and executed according to the specification [144]. The technical nuances that can be introduced by different implementation of the same concept, makes the auditing process more complicated and requires more technical expertise on the specific cryptoasset, smart contract, and the blockchain under audit.

Paths Forward
Reject Cryptoassets Audits
Collaborate with Experts
Develop In-house Expertise
Maturity of Cryptoassets (Test of Time)
Precedence of Previous Audits

Figure 6.4: Paths Forward for Auditing Cryptoassets

As we pointed out in the previous section, there are standards, such as the C4 Cryptoasset Security Standard Audit [57], that can be used by the auditors to ensure the adequacy of the internal controls of the cryptoassets. However, as noted, the audit of the cryptoassets require more technical knowledge of the field than previously required on traditional audits, where relying on third party statements were sufficient. This is as if every auditor requires to become a blockchain (smart contract) security auditor as the task of auditing the cryptoassets requires verifying the technical details of the implementation and the code.

6.5.1 Paths Forward

We will now look at the different paths forward that auditing firms have taken in the past [261, 145], and how it is evolving over time with the maturity of the ecosystem.

Reject cryptoasset audits: auditing firms can *reject cryptoasset audits* as they have done in early days due to their unknown risk models. This was the case with the early days of the internet, where the auditing firms were not comfortable with the risk models of the internet companies, and rejected the audits. Similarly, many early companies trying to get audited, were rejected by the auditing firms, as they were not comfortable with the risk models of the cryptoassets, and it's been ongoing since. One recent example is Binance, one of the biggest cryptocurrency exchanges, was rejected by the auditing firms in 2022 [59], to conduct a proof-of-reserves audit [95].

Collaborate with experts: as it has been the case, audit firms can *collaborate with experts* in the field to sign off on the risk assessment of the cryptoassets in question. There is significant research and development in the field of blockchain security and smart contract audits, which can be utilized to ensure the adequacy of the internal controls and work flow of each cryptoasset [187]. Additionally, methods such as formal verification [73], can be used to ensure the correctness of the smart contracts and the implementation of the cryptoassets underlying technology.

Develop their own expertise in-house: the auditing firms started to *develop their own expertise in-house* and create their own standards, tools, and procedures for auditing the cryptoassets [253, 3]. This is a challenging task, as it requires significant investment in the field (R&D), and the auditing firms must be able to attract and retain the talent required for such task. Many of the blockchain security firms have hard time finding talent, and the auditing firms are facing the same challenge.

Test of time: the *maturity* of the cryptoassets themselves can result in more confidence in the security of the underlying technology (Test of Time), such as the case with Bitcoin and Ethereum. As the cryptoassets mature in the market capitalization, demand, development, and the underlying technology, more banks [20, 245], regulators [247, 322], and auditing firms [186] started to recognize the importance of the cryptoassets and started to research further into the field [79, 297].

Rely of precedence of previous audits: As more companies are holding cryptoassets, and more audit firms look into these cryptoassets, the auditing process will become more standardized and can be built on top of the previous experiences and reports [88, 154]. Integrating standards such as the C4 Cryptoasset Security Standard Audit [57] and security audit procedures, can result in more efficient auditing process for the future audits.

6.6 Conclusion

This chapter has aimed to demonstrate that, in comparison to traditional audits, audits of clients who hold material amounts of cryptoassets are complex but not impossible. Once a client has been accepted, the three most cited stumbling blocks to providing an audit opinion are the existence, ownership, and valuation of cryptoassets. However, we argue that these issues are not insurmountable if industry guidelines are put into place to allow auditors to verify their client's cryptographic keys against a "state of the world" at a generally accepted point in time. Verifying existence and ownership largely hinges on an auditor's ability to verify the possession of cryptographic keys. However, the auditor must be certain that these keys in fact belong to the client and do not simply represent access to an account. Once ownership has been proven, the auditor can rely on the immutable properties of the blockchain to verify existence as the blockchain provides the entire record of transactions since the blockchain's inception. The issue of key sharing is important but is not unlike a situation in the real world where a related party could give the entity under audit a large sum of cash to hold at year-end and report on their financial statement to buoy their financial performance. Volatility complicates the valuation of altcoins and other coins with low trading volumes. However, many other exotic securities exist where accountants rely on complex financial modeling to determine a price.

In sum, this chapter argues that although auditors are rightly cautious when approaching a new sector where clients have not been initiated to the importance of internal controls and where numerous frauds have recently been perpetrated, audits are possible. Additionally, by collaborating closely with blockchain experts and security auditors, auditors who themselves are not experts in blockchain technology can come to a place where they can provide assurance to this sector.

Chapter 7

Concluding Remarks

The design of the blockchain technology and the applications of it can enable really novel approaches to remove trust in the intermediaries and significantly change the information flow in different businesses. However, this also brings forth some unforeseen consequences that were not possible before the existence of this technology.

In this thesis, we have explored some aspects of the blockchain technology and its applications, which as a consequence of the design and implementation of the technology, can lead to some unforeseen consequences. We have explored the following topics in this thesis:

- Chapter 3: **Cryptojacking: from Replacing Ads to Invisible Abuse**: Browser-based mining promised to replace ads as a source of revenue for websites. However, it was abused by malicious actors and became a threat to the users. We have explored the different aspects of this threat in the wild and discussed possible solutions and ethical issues that might arise from the solutions.

- Chapter 4: **Blockchain Front-running: from Transparency to Extracting Value**: The blockchain technology promises to bring transparency to the financial markets. However, the transparency and permissionless aspect of the blockchains enables many new actors to observe the information flow and attempt to extract value from it. We showcase how some of these attacks happened in the past and discuss possible solutions to mitigate them. Furthermore, we discuss the ethical and legal issues that are left unanswered.

- Chapter 5 **Oracles: from Ground Truth to Market Manipulation**: A blockchain, even though exists in an online decentralized network, it is still a closed system and cannot access the information from the outside world. Oracles are the entities that provide this information to the blockchain. However, the design of the oracles can lead to some unforeseen consequences and attacks that are foundational to the decentralized applications. We discuss the different types of oracles designs and implementations, and how they can be manipulated by malicious actors.

- Chapter 6 **Blockchain Audits: from Existence to Internal Controls**: Cryptoassets are assets that are mainly defined be some code that lives on the blockchain. The code is open source and anyone can read it. However, the code is not the only thing that matters in the financial statements of the company holding those cryptoassets. We discuss how auditors need to look at these assets, to be able to identify their existence and valuation, how the ownership can be proven and what it means to look at the internal control of the cryptoassets. Further we discuss some challenges and potential solutions to these problems.

Each chapter in this thesis discusses their own future work and related *follow-up work*. However, there are some overarching themes that can be explored further.

7.1 Future Work

As we discussed in details in each chapter, there are many open questions and challenges that need to be addressed. Some of these challenges are technical in nature, some regarding user education on how to think about this new technology [144], while others are more related to the legal and ethical aspects of the blockchain technology integration with the current systems. In this section, we shortly discuss some of the future work.

Online advertisement is a multi-billion dollar industry. The current online advertisement ecosystem is broken, and it is not clear how to fix it. The current solutions are either not user-friendly or not privacy-preserving. Malvertising [196] is a big threat to the users and given its complicated web of actors, it is almost impossible to change the current ecosystem. However, with blockchain technology we can change this ecosystem and bring transparency and fairness to the users. We have discussed some of the challenges and possible solutions in Chapter 3. However, there are many open questions that need to be addressed. For example, how to design a system that is fair to the users and the publishers, while at the same time it is not vulnerable to the malicious actors? What would the user consent look like?, and many more.

As we further explained in Chapter 4 and Chapter 5, the blockchain technology promises to bring transparency to the financial markets. However, the transparency and permissionless aspect of the blockchains enables many new actors to observe the information flow and attempt to extract value from it. Additionally, due to the close nature of blockchains, oracles are needed to bring the information from the outside world to the blockchain, and can be used to manipulate the market, either maliciously or unintentionally due to a technical bug. We discussed some potential solutions for front-running, which one of the most popular ones so far has been to embrace the front-running but "democratize access" to it by sharing the profit within the participating actors. This solution has gained a lot of traction in the blockchain community, however, it is not clear how to implement it in a way that is fair to all the participants. Furthermore, as we explored the early days of front-running in the traditional financial markets, we saw that the regulators were not able to keep up with the pace of the technology and the market. Similarly, It is not clear how the regulators can keep up with the pace of the MEV and the technical nuances of this new aspect of the blockchain technology.

Following the same pattern as the case studies in Chapter 6, there are many new possible approaches for financial statements, such as Real-time Financial Reporting, that were not possible to implement in a trust-minimized way before (See Section 6.4.3). Although technically feasible, there are many nuances that changes the way auditors can trust and verify the information. Furthermore, the current accounting standards are not designed for this new technology, and it is not clear how to apply them to the blockchain technology, there are many examples in the chapter. Additionally, we still discussed a centralized company holding cryptoassets, however, many new entities in the ecosystem are DAOs (Decentralized Autonomous Organizations) that are not controlled by a centralized entity [304]. It is not clear how to audit these entities and what it means to audit these entities.

I hope this thesis can be a starting point for further research in this area. I believe that the blockchain technology can bring a lot of benefits to the society, however, it requires further research on ethical and fair design of the systems for all the participants that also does not lead to malicious unforeseen consequences.

Bibliography

[1] P. W. Abreu, M. Aparicio, and C. J. Costa. Blockchain technology in the auditing environment. In *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6. IEEE, 2018.

[2] M. E. Acer, E. Stark, A. P. Felt, S. Fahl, R. Bhargava, B. Dev, M. Braithwaite, R. Sleevi, and P. Tabriz. Where the wild warnings are: Root causes of chrome https certificate errors. In *CCS*, CCS '17, pages 1407–1420, New York, NY, USA, 2017. ACM.

[3] S. F. Adam Regelbrugge. An internal auditors guide to auditing blockchain. *Deloitte*, 2022.

[4] C. Adams and S. Lloyd. *Understanding public-key infrastructure: concepts, standards, and deployment considerations.* Sams Publishing, 1999.

[5] H. Adams. Uniswap. *URl: https://uniswap.org/docs*, 2019.

[6] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, and D. Robinson. Uniswap v3 core. 2021.

[7] J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira, and A. Kastania. Astraea: A decentralized blockchain oracle. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1145–1152. IEEE, 2018.

[8] M. Alizadeh, M. Zamani, A. R. Shahemabad, J. Shayan, and A. Azarnik. A survey on attacks in rfid networks. *Open International Journal of Informatics (OIJI)*, 1(1):15–24, 2012.

[9] D. W. Allen, C. Berg, and A. M. Lane. Why airdrop cryptocurrency tokens? *Journal of Business Research*, 163:113945, 2023.

[10] E. Anceaume, A. Pozzo, T. Rieutord, and S. Tucci-Piergiovanni. On finality in blockchains. *arXiv preprint arXiv:2012.10172*, 2020.

[11] G. Angeris and T. Chitra. Improved price oracles: Constant function market makers. *arXiv preprint arXiv:2003.10001*, 2020.

[12] Anonymous. How the first winner of fomo3d won the jackpot? `https://winnerfomo3d.home.blog/`, 2018. Accessed: 2018-09-09.

[13] Aragon. Build better, together - aragon dao.

[14] A. Asayag, G. Cohen, I. Grayevsky, M. Leshkowitz, O. Rottenstreich, R. Tamari, and D. Yakira. A fair consensus protocol for transaction ordering. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pages 55–65. IEEE, 2018.

[15] N. Atzei, M. Bartoletti, and T. Cimoli. A survey of attacks on ethereum smart contracts (sok). In *International conference on principles of security and trust*, pages 164–186. Springer, 2017.

[16] R. Aune, M. O'Hara, and O. Slama. Footprints on the blockchain: Information leakage in distributed ledgers. *Available at SSRN 2896803*, 2017.

[17] F. I. R. Authority. Im-2110-3. front running policy, 2002.

[18] F. I. R. Authority. 5270. front running of block transactions, 2012.

[19] F. I. R. Authority. Notice of filing of proposed rule change to adopt finra rule 5270 (front running of block transactions) in the consolidated finra rulebook, 2012.

[20] P. D. Azar, G. Baughman, F. Carapella, J. Gerszten, A. Lubis, J. P. Perez-Sangimino, C. Scotti, N. Swem, A. Vardoulakis, and D. E. Rappoport W. The financial stability implications of digital assets. *FRB of New York Staff Report*, (1034), 2022.

[21] K. M. Bakarich, J. Castonguay, and P. E. OB́rien. The use of blockchains to enhance sustainability reporting and assurance. *Accounting Perspectives*, 19(4):389–412, 2020.

[22] Balancer. A protocol for programmable liquidity. *URl: https://balancer.finance/*, 2020.

[23] T. Bamert, C. Decker, L. Elsen, R. Wattenhofer, and S. Welten. Have a snack, pay with bitcoins. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pages 1–5. IEEE, 2013.

[24] T. D. Bauer, C. Estep, and B. Malsch. One team or two? investigating relationship quality between auditors and it specialists: Implications for audit team identity and the audit process. *Contemporary Accounting Research*, 36(4):2142–2177, 2019.

[25] C. Baum, J. Hsin-yu Chiang, B. David, T. K. Frederiksen, and L. Gentile. Sok: Mitigation of front-running in decentralized finance. In *International Conference on Financial Cryptography and Data Security*, pages 250–271. Springer, 2022.

[26] BBC. Websites hacked to mint crypto-cash. `http://www.bbc.com/news/technology-41518351`, 2017.

[27] BDO. Independent third-party reviews (itpr). online, 2019.

[28] D. Beaver and S. Haber. Cryptographic protocols provably secure against dynamic adversaries. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 307–323. Springer, 1992.

[29] J. Bebel and D. Ojha. Ferveo: Threshold decryption for mempool privacy in bft networks. *Cryptology ePrint Archive*, 2022.

[30] B. Benligiray, S. Milic, and H. Vänttinen. Decentralized apis for web 3.0.

[31] T. Bernani. Oraclize. *URl: https://www.oraclize.it/*, 2016.

[32] H. L. Bijmans, T. M. Booij, and C. Doerr. Inadvertently making cyber criminals rich: A comprehensive study of cryptojacking campaigns at internet scale. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1627–1644, 2019.

[33] A. Biryukov, D. Khovratovich, and S. Tikhomirov. Findel: Secure derivative contracts for ethereum. In *International Conference on Financial Cryptography and Data Security*, pages 453–467. Springer, 2017.

[34] G. Bissias, B. N. Levine, A. P. Ozisik, and G. Andresen. An analysis of attacks on blockchain consensus. *arXiv preprint arXiv:1610.07985*, 2016.

[35] Bitaccess. Canadian government launches on ethereum blockchain. *Yahoo! Finance*, 2018. URl: https://finance.yahoo.com/news/canadian-government-launches-ethereum-block-211507375.html.

[36] BleepingComputer. The internet is ride with in-browser miners and it is getting worse each day. `https://www.bleepingcomputer.com/news/security/the-internet-is-rife-with-in-browser-miners-and-its-getting-worse-each-day/`, 2012. Accessed: 2017-12-08.

[37] C. Blog. 16 ways to create dynamic non-fungible tokens (nft) using chainlink oracles. *URl: https://blog.chain.link/create-dynamic-nfts-using-chainlink-oracles/*, 2021.

[38] I. Bogatyy. Implementing ethereum trading front-runs on the bancor exchange in python. 2017. (Accessed on 08/13/2018).

[39] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In *CRYPTO*, 2018.

[40] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In *Advances in Cryptology – CRYPTO 2018*, volume 10991 of *Lecture Notes in Computer Science*, pages 757–788. Springer, 2018.

[41] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.

[42] J. Bonneau, E. W. Felten, S. Goldfeder, J. A. Kroll, and A. Narayanan. Why buy when you can rent? bribery attacks on bitcoin consensus. 2016.

[43] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[44] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A.-R. Sadeghi. Software grand exposure:{SGX} cache attacks are practical. In *11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17)*, 2017.

[45] L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz, et al. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. *Chainlink Labs*, 1:1–136, 2021.

[46] L. Breidenbach, C. Cachin, A. Coventry, A. Juels, and A. Miller. Chainlink off-chain reporting protocol. *URl: https://blog.chain.link/off-chain-reporting-live-on-mainnet/*, 2021.

[47] L. Breidenbach, I. Cornell Tech, P. Daian, F. Tramer, and A. Juels. Enter the hydra: Towards principled bug bounties and exploit-resistant smart contracts. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*. {USENIX} Association, 2018.

[48] L. Breidenbach, P. Daian, A. Juels, and F. Tramer. To sink frontrunners, send in the submarines. `http://hackingdistributed.com/2017/08/28/submarine-sends/`, 2017. Accessed: 2018-08-28.

[49] L. Breidenbach, T. Kell, S. Gosselin, and S. Eskandari. Libsubmarine: Defeat front-running on ethereum. `https://libsubmarine.org/`, 2018. Accessed: 2018-12-07.

[50] D. Broby and G. Paul. The financial auditing of distributed ledgers, blockchain and cryptocurrencies. *Journal of Financial Transformation*, 46:76–88, 2017.

[51] J. V. Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx. Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution. In

USENIX Security Symposium, Baltimore, MD, Aug. 2018. USENIX Association.

[52] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 319–338, 2018.

[53] B. Bünz, S. Goldfeder, and J. Bonneau. Proofs-of-delay and randomness beacons in ethereum. *IEEE Security and Privacy on the blockchain (IEEE S&B)*, 2017.

[54] V. Buterin. Uni should become an oracle token. *URl: https://gov.uniswap.org/t/uni-should-become-an-oracle-token/11988*, 2021.

[55] V. Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3:37, 2014.

[56] S. Buti, B. Rindi, and I. M. Werner. Diving into dark pools. 2011.

[57] C. C. C. (C4). (ccssa) crypto currency security standard auditor, 2023. URl: https://cryptoconsortium.org/certifications/ccssa/.

[58] M. Caprolu, S. Raponi, G. Oligeri, and R. Di Pietro. Cryptomining makes noise: Detecting cryptojacking via machine learning. *Computer Communications*, 171:126–139, 2021.

[59] B. S. CASEY WAGNER. As crypto auditors call it quits, what will take their place?, 2022.

[60] M. Castro and B. Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002.

[61] C. CGMA. Blockchain augmented audit–benefits and challenges for accounting professionals. *The journal of theoretical accounting research*, 14(1):117–137, 2018.

[62] Chainlink. Chainlink vrf: On-chain verifiable randomness. *URl: https://blog.chain.link/verifiable-random-functions-vrf-random-number-generation-rng-feature/*, 2020.

[63] ChainLink. What is the blockchain oracle problem. *URl: https://blog.chain.link/what-is-the-blockchain-oracle-problem/*, 2020.

[64] K. Chatterjee, A. K. Goharshady, and A. Pourdamghani. Probabilistic smart contracts: Secure randomness on the blockchain. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 403–412. IEEE, 2019.

[65] CheckPointResearchTeam. October's most wanted malware: Cryptocurrency mining presents new threat. https://blog.checkpoint.com/2017/11/13/octobers-wanted-malware-cryptocurrency-mining-presents-new-threat/, 2017.

[66] H. Chen, M. Pendleton, L. Njilla, and S. Xu. A survey on ethereum systems security: Vulnerabilities, attacks, and defenses. *ACM Computing Surveys (CSUR)*, 53(3):1–43, 2020.

[67] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contract execution. *arXiv preprint arXiv:1804.05141*, 2018.

[68] M. Ciampi, M. Ishaq, M. Magdon-Ismail, R. Ostrovsky, and V. Zikas. Fairmm: A fast and frontrunning-resistant crypto market-maker. In *International Symposium on Cyber Security, Cryptology, and Machine Learning*, pages 428–446. Springer, 2022.

[69] CK. Secret markets: Crypto front running prevention for automated market makers (amms). https://scrt.network/blog/secret-markets-front-running-prevention, 2020. Accessed: 2023-12-25.

[70] J. Clark, J. Bonneau, E. W. Felten, J. A. Kroll, A. Miller, and A. Narayanan. On decentralizing prediction markets and order books. In *Workshop on the Economics of Information Security, State College, Pennsylvania*, 2014.

[71] J. Clark, D. Demirag, and S. Moosavi. Sok: Demystifying stablecoins. *CACM*, 2020.

[72] J. Clark, H. Dhamija, A. Faz-Hernández, A. Gervais, K. Gudgeon, and I. Miers. Sok: Layer-two blockchain protocols. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019.

[73] J. Clark, E. C. Miller, and S. J. Smith. Sok: Towards formal verification of smart contracts. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 783–800. IEEE, 2018.

[74] J. Clark and P. C. Van Oorschot. Sok: Ssl and https: Revisiting past challenges and evaluating certificate trust model enhancements. In *2013 IEEE Symposium on Security and Privacy*, pages 511–525. IEEE, 2013.

[75] CoinDesk. State of blockchains q3 2018, 2018.

[76] Coinhive. Coinhive monetize your business with your users cpu power. https://coinhive.com/, 2017. Accessed: 2017-11-20.

[77] CoinMarketCap. Total market capitalization., 2023.

[78] Coinmonk. Vulnerabilities in maker: Oracle-governance attacks, attack daos, and (de)centralization. *URl: https://medium.com/coinmonks/vulnerabilities-in-maker-oracle-governance-attacks-attack-daos-and-de-centralization-d943685adc2f*, 2019.

[79] B. Committee et al. Prudential treatment of cryptoasset exposures. *Basel Committee on Banking Supervision, BIS Consultative Document*, 2021.

[80] I. A. Committee. Ssac advisory on domain name front running. 10 2007. (Accessed on 08/15/2018).

[81] Compound. Open price feed. *URl: https://compound.finance/docs/prices*, 2020.

[82] Consensys. Ethereum smart contract best practices, 2020.

[83] ConsenSys-Diligence. Security review of 0x smart contracts, 2017.

[84] M. Conti, A. Gangwal, and S. Ruj. On the economic significance of ransomware campaigns: A bitcoin transactions perspective. *Computers & Security*, 79:162–189, 2018.

[85] T. Core. Staking, disputes, and voting. *URl: https://medium.com/tellor/staking-disputes-and-voting-ad09c66eb7bc/*, 2019.

[86] V. Costan and S. Devadas. Intel sgx explained. *IACR Cryptol. ePrint Arch.*, 2016(86):1–118, 2016.

[87] J. G. Coyne and P. L. McMickle. Can blockchains serve an accounting purpose? *Journal of emerging technologies in accounting*, 14(2):101–111, 2017.

[88] T. C. P. A. B. (CPAB). Auditing in the crypto-asset sector. *CPAB EXCHANGE*, 2022.

[89] M. Crosby, P. Pattanayak, S. Verma, V. Kalyanaraman, et al. Blockchain technology: Beyond bitcoin. *Applied Innovation*, 2(6-10):71, 2016.

[90] T. B. Crypto. Chainlink nodes were targeted in an attack last weekend that cost them at least 700 eth. *URl: https://www.theblockcrypto.com/post/76986/chainlink-nodes-attack-eth*, 2020.

[91] T. B. Crypto. Defi attacks: the general picture. *URl: https://www.theblockcrypto.com/research/105472/defi-attacks-the-general-picture*, 2021.

[92] Cryptonote. Cryptonote technology. `https://cryptonote.org/inside.php#equal-proof-of-work`, 2017. Accessed: 2017-11-20.

[93] M. B. Curtis, J. G. Jenkins, J. C. Bedard, and D. R. Deis. Auditors' training and proficiency in information systems: A research synthesis. *Journal of information systems*, 23(1):79–96, 2009.

[94] P. M. da Silva, M. Matos, and J. Barreto. Fixed transaction ordering and admission in blockchains. *EuroDW2018/papers/eurodw18-Silva. pdf*, 2019.

[95] G. G. Dagher, B. Bünz, J. Bonneau, J. Clark, and D. Boneh. Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 720–731, 2015.

[96] J. Dai and M. A. Vasarhelyi. Toward blockchain-based accounting and assurance. *Journal of information systems*, 31(3):5–21, 2017.

[97] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 910–927. IEEE, 2020.

[98] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. In *IEEE Symposium on Security and Privacy*, 2020.

[99] P. Daian, T. Kell, I. Miers, and A. Juels. On-chain vote buying and the rise of dark daos. *URl: https://hackingdistributed.com/2018/07/02/on-chain-vote-buying/*, 2018.

[100] P. Daian, R. K. Knechtel, J. Clark, D. J. Wang, S. Bowe, P. Berman, and A. Miller. Flash boys 2.0: Frontrunning, transaction rebundling, and other flash boys in decentralized exchanges. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019.

[101] Decrypt. Oracle exploit sees $89 million liquidated on compound. *URl: https://decrypt.co/49657/oracle-exploit-sees-100-million-liquidated-on-compound*, 2020.

[102] DeepDotWeb. Coinhive hacked and launches new opt-in service. `https://www.deepdotweb.com/2017/11/11/coinhive-hacked-launches-new-opt-service/`, 2017.

[103] A. des marchés financiers. Bulletin: Interdictions. 15 (36). 2018.

[104] E. devp2p. Ethereum wire protocol (eth). `https://github.com/ethereum/devp2p/blob/master/caps/eth.md`, 2020. Accessed: 2023-12-18.

[105] DIA. Dia — transparent oracles for a decentralised financial ecosystem, 2020.

[106] E. Discussion. Handling frontrunning in the permanent registrar. 2018.

[107] distribuyed. A comprehensive list of decentralized exchanges (dex) of cryptocurrencies, tokens, derivatives and futures, and their protocols. `https://distribuyed.github.io/index/`, 2018. Accessed: 2018-09-24.

[108] DODO. The dodo advantage. *URl: https://dodoex.github.io/docs/docs/advantages*, 2020.

[109] J. R. Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

[110] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman. A search engine backed by Internet-wide scanning. In *ACM CCS*, Oct. 2015.

[111] B. Edelman. Front-running study: Testing report. 2009.

[112] S. Ellis, A. Juels, and S. Nazarov. Chainlink: A decentralized oracle network. *Retrieved March*, 11:2018, 2017.

[113] W. Entriken, D. Shirley, J. Evans, and N. Sachs. Erc-721 non-fungible token standard. `https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md`, 2018. Accessed: 2018-08-31.

[114] S. Eskandari. Real-world deployability and usability of bitcoin. 2015. Available at `https://spectrum.library.concordia.ca/id/eprint/980859/`.

[115] S. Eskandari. Ethereum name service audit. *ConsenSys Diligence*, 2019. URl: https://consensys.io/diligence/blog/2019/04/ethereum-name-service-audit/.

[116] S. Eskandari. Etherroll bug thread. *URl: https://twitter.com/sbetamc/status/1263220679937265671*, 2020.

[117] S. Eskandari. Token interaction checklist. *ConsenSys Diligence*, 2020. URl: https://consensys.io/diligence/blog/2020/11/token-interaction-checklist/.

[118] S. Eskandari, D. Barrera, E. Stobert, and J. Clark. A first look at the usability of Bitcoin key management. In *USEC*, 2015.

[119] S. Eskandari, J. Clark, and A. Hamou-Lhadj. Buy your coffee with bitcoin: Real-world deployment of a bitcoin point of sale terminal. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 382–389. IEEE, 2016.

[120] S. Eskandari, J. Clark, V. Sundaresan, and M. Adham. On the feasibility of decentralized derivatives markets. In *International Conference on Financial Cryptography and Data Security*, pages 553–567. Springer, 2017.

[121] S. Eskandari, A. Leoutsarakos, T. Mursch, and J. Clark. A first look at browser-based cryptojacking. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 58–66. IEEE, 2018.

[122] S. Eskandari, S. Moosavi, and J. Clark. Sok: Transparent dishonesty: front-running attacks on blockchain. In *WTSC*. FC, 2019.

[123] S. Eskandari, M. Salehi, W. C. Gu, and J. Clark. Sok: oracles from the ground truth to market manipulation. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, pages 127–141, 2021.

[124] Ethereum. Official go implementation of the ethereum protocol, 2015.

[125] Ethereum. Go-ethereum: worker.go - commitnewwork(). 2018. Accessed: 2018-12-07.

[126] Ethereum. What was the merge? `https://ethereum.org/en/roadmap/merge/`, 2022. Accessed: 2023-12-18.

[127] Ethereum. Gas and fees, 2023. URl: https://ethereum.org/en/developers/docs/gas/.

[128] Ethereum. Transactions, 2023. URl: https://ethereum.org/en/developers/docs/transactions/.

[129] Ethereum.org. Account types, gas, and transactions. ethereum homestead 0.1 documentation. `https://ethereum.org/en/developers/docs/accounts#what-is-gas`. (Accessed on 06/14/2018).

[130] Ethereum.org. Opcodes for the evm. *Ethereum Website*, 2020. URl: https://ethereum.org/developers/docs/evm/opcodes.

[131] Etheroll. Ethereum casino. *URl: https://etheroll.com/*, 2020.

[132] European-Commission. Cookies. `http://ec.europa.eu/ipg/basics/legal/cookies/index_en.htm`, 2011. Accessed: 2017-12-08.

[133] ExtremeTech. Browser-based mining malware found on pirate bay, others. `https://www.extremetech.com/internet/255971-browser-based-cryptocurrency-malware-appears-online-pirate-bay`, 2017. Accessed: 2017-11-20.

[134] EY. The valuation of crypto-assets. *Ernst & Young*, 2019.

[135] I. Eyal. On cryptocurrency wallet design. Cryptology ePrint Archive, Paper 2021/1522, 2021. `https://eprint.iacr.org/2021/1522`.

[136] H. Feistel. Cryptography and computer privacy. *Scientific american*, 228(5):15–23, 1973.

[137] FinancialTimes. Barclays trader charged with front-running by us authorities. 2018.

[138] A. Firat, S. Madnick, and B. Grosof. Knowledge integration to overcome ontological heterogeneity: Challenges from financial information systems. 2002.

[139] flashbots. Flashbots is a research and development organization formed to mitigate the negative externalities posed by maximal extractable value (mev) to stateful blockchains, starting with ethereum., 2020. URL: https://www.flashbots.net/.

[140] Flashbots. Flashbots. *URl: https://github.com/flashbots/pm*, 2021.

[141] B. Ford and R. Böhme. Rationality is self-defeating in permissionless systems. 2019.

[142] Fortune. Popular google chrome extension caught mining cryptocurrency on thousands of computers. `http://fortune.com/2018/01/02/google-chrome-extension-cryptocurrency-mining-monero/`, 2017. Accessed: 2018-01-20.

[143] M. Forum. Scientific governance and risk #120. *URl: https://forum.makerdao.com/t/agenda-discussion-scientific-governance-and-risk-120-thursday-december-3-16-00-utc/5357*, 2020.

[144] A. Gaggioli, S. Eskandari, P. Cipresso, and E. Lozza. The middleman is dead, long live the middleman: The "trust factor" and the psycho-social implications of blockchain. *Frontiers in Blockchain*, 2:20, 2019.

[145] T. Garanina, M. Ranta, and J. Dumay. Blockchain in accounting research: current trends and emerging topics. *Accounting, Auditing & Accountability Journal*, 35(7):1507–1533, 2022.

[146] J. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In *EUROCRYPT*, 2015.

[147] S. Goldberg, D. Papadopoulos, and J. Včelák. Verifiable Random Functions (VRFs). Internet-Draft draft-goldbe-vrf-01, Internet Engineering Task Force, June 2017. Work in Progress.

[148] F. Gomes and M. Correia. Cryptojacking detection with cpu usage metrics. In *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, pages 1–10. IEEE, 2020.

[149] K. Govindarajan, D. Vinayagamurthy, P. Jayachandran, and C. Rebeiro. Privacy-preserving decentralized exchange marketplaces. In *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9. IEEE, 2022.

[150] V. Gramoli. From blockchain consensus back to byzantine consensus. *Future Generation Computer Systems*, 107:760–769, 2020.

[151] I. Grigg. Triple entry accounting. *Systemics Inc*, pages 1–10, 2005.

[152] W. C. Gu, A. Raghuvanshi, and D. Boneh. Empirical measurements on pricing oracles and decentralized governance for stablecoins. *Available at SSRN 3611231*, 2020.

[153] F. Hakimpour and A. Geppert. Resolving semantic heterogeneity in schema integration. In *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, pages 297–308, 2001.

[154] H. Han, R. K. Shiwakoti, R. Jarvis, C. Mordi, and D. Botchie. Accounting and auditing with blockchain technology and artificial intelligence: A literature review. *International Journal of Accounting Information Systems*, 48:100598, 2023.

[155] R. Hanson. Combinatorial information market design. *Information Systems Frontiers*, 5(1):107–119, 2003.

[156] E. Heilman, A. Kendler, and A. Zohar. Eclipse attacks on bitcoin's peer-to-peer network. In *24th USENIX Security Symposium (USENIX Security 15)*, 2015.

[157] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg. Eclipse attacks on bitcoins peer-to-peer network. In *USENIX Security*, pages 129–144, Washington, D.C., 2015. USENIX Association.

[158] L. Heimbach, L. Kiffer, C. Ferreira Torres, and R. Wattenhofer. Ethereum's proposer-builder separation: Promises and realities. In *Proceedings of the 2023 ACM on Internet Measurement Conference*, pages 406–420, 2023.

[159] L. Heimbach and R. Wattenhofer. Sok: Preventing transaction reordering manipulations in decentralized finance. *arXiv preprint arXiv:2203.11520*, 2022.

[160] S. Henningsen, D. Teunis, M. Florian, and B. Scheuermann. Eclipsing ethereum peers with false friends. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 300–309. IEEE, 2019.

[161] C. Herley and P. C. Van Oorschot. Sok: Science, security and the elusive goal of security as a scientific pursuit. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 99–120, 2017.

[162] E. Hertzog, G. Benartzi, and G. Benartzi. Bancor protocol: continuous liquidity for cryptographic tokens through their smart contracts. *White paper*, 2017.

[163] R. M. Hirsch. The effect of spatial distance and domain knowledge distinctiveness on auditor reliance on it specialists. *Journal of Information Systems*, 34(1):81–103, 2020.

[164] G. Hong, Z. Yang, S. Yang, L. Zhang, Y. Nan, Z. Zhang, M. Yang, Y. Zhang, Z. Qian, and H. Duan. How you get shot in the back: A systematical study about cryptojacking in the real world. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1701–1713, 2018.

[165] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox. Zcash protocol specification. Technical report, Technical report, 2016–1.10. Zerocoin Electric Coin Company, 2016.

[166] D. Y. Huang, H. Dharmdasani, S. Meiklejohn, V. Dave, C. Grier, D. McCoy, S. Savage, N. Weaver, A. C. Snoeren, and K. Levchenko. Botcoin: Monetizing stolen cycles. In *NDSS*, 2014.

[167] initc3.org. Frontrun me. *URL: http://frontrun.me/*, 2018.

[168] H. D. Insight. Price oracle - a must have infrastructure. *URl: https://www.huobidefilabs.io/en/Insight/1313163603254243330/*, 2020.

[169] International Accounting Standards Board. Ifrs 13 — fair value measurement, 2013.

[170] G. Issue. Method ‘decreaseapproval‘ in unsafe. 2017.

[171] N. Johnson. Ethereum domain name service - specification. 2016.

[172] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten. Arbitrum: Scalable, private smart contracts. In *USENIX Security Symposium*, pages 1353–1370, 2018.

[173] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten. Arbitrum: Scalable, private smart contracts. In *USENIX Security*, 2018.

[174] H. A. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan. An empirical study of namecoin and lessons for decentralized namespace design. In *WEIS*. Citeseer, 2015.

[175] G. Kappos, H. Yousaf, M. Maller, and S. Meiklejohn. An empirical analysis of anonymity in zcash. *arXiv preprint arXiv:1805.03180*, 2018.

[176] G. O. Karame, E. Androulaki, and S. Capkun. Double-spending fast payments in bitcoin. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 906–917. ACM, 2012.

[177] M. Kelkar, S. Deb, S. Long, A. Juels, and S. Kannan. Themis: Fast, strong order-fairness in byzantine consensus. Cryptology ePrint Archive, Paper 2021/1465, 2021. `https://eprint.iacr.org/2021/1465`.

[178] M. Kelkar, F. Zhang, S. Goldfeder, and A. Juels. Order-fairness for byzantine consensus. In *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part III 40*, pages 451–480. Springer, 2020.

[179] A. Kharraz, Z. Ma, P. Murley, C. Lever, J. Mason, A. Miller, N. Borisov, M. Antonakakis, and M. Bailey. Outguard: Detecting in-browser covert cryptocurrency mining in the wild. In *The World Wide Web Conference*, pages 840–852, 2019.

[180] D. Khovratovich and J. Law. Sovrin: digital identities in the blockchain era. Technical report, Sovrin, 2017.

[181] M. King, J. Atkins, and M. Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *The American economic review*, 97(1):242–259, 2007.

[182] M. B. Koch. Exploring cryptokitties part 2 - the cryptomidwives. 2018.

[183] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom. Spectre attacks: Exploiting speculative execution. In *IEEE Symposium on Security and Privacy*, pages 1–19, 2019.

[184] R. K. Konoth, E. Vineti, V. Moonsamy, M. Lindorfer, C. Kruegel, H. Bos, and G. Vigna. Minesweeper: An in-depth look into drive-by cryptocurrency mining and its defense. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1714–1730, 2018.

[185] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE symposium on security and privacy (SP)*, pages 839–858. IEEE, 2016.

[186] KPMG. The rise of cryptoasset adoption in canada, 2022.

[187] Kraken. Proof of reserves or proof of nothing: There is no in between. 2022.

[188] A. Kroeger and A. Sarkar. The law of one bitcoin price. *Federal Reserve Bank of Philadelphia*, 2017.

[189] J. A. Kroll, I. C. Davey, and E. W. Felten. The economics of bitcoin mining, or bitcoin in the presence of adversaries. In *Proceedings of WEIS*, volume 2013, page 11, 2013.

[190] A. Kumar, C. Fischer, S. Tople, and P. Saxena. A traceability analysis of monero's blockchain. In *European Symposium on Research in Computer Security*, pages 153–173. Springer, 2017.

[191] H. Lambur, A. Lu, and R. Cai. Uma data verification mechanism: Adding economic guarantees to blockchain oracles. *Risk Labs, Inc., Tech. Rep., Jul*, 2019.

[192] Langner. The stuxnet story. *URl: https://www.langner.com/2020/07/the-stuxnet-story/*, 2020.

[193] R. Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51, 2011.

[194] C. Lesaege, F. Ast, and W. George. Kleros the justice protocol. *URl: https://kleros.io/static/whitepaper_en-8bd3a0480b45c39899787e17049ded26.pdf*, 2019.

[195] Z. Li, J. Li, Z. He, X. Luo, T. Wang, X. Ni, W. Yang, X. Chen, and T. Chen. Demystifying defi mev activities in flashbots bundle. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 165–179, 2023.

[196] Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang. Knowing your enemy: understanding and detecting malicious web advertising. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 674–686, 2012.

[197] S. Libs. Veedo a stark-based verifiable delay function, 2020.

[198] I.-C. Lin and T.-C. Liao. A survey of blockchain security issues and challenges. *IJ Network Security*, 19(5):653–659, 2017.

[199] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg. Meltdown: Reading kernel memory from user space. In *USENIX Security Symposium*, pages 973–990, Baltimore, MD, Aug. 2018. USENIX Association.

[200] J. Liu, Z. Zhao, X. Cui, Z. Wang, and Q. Liu. A novel approach for detecting browser-based silent miner. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pages 490–497. IEEE, 2018.

[201] LiveHelpNow. Security incident nov 23rd, 2017. `https://blog.livehelpnow.net/security-incident-nov-23rd-2017/`, 2017. Accessed: 2017-12-14.

[202] G. T. LLP. Managing virtual currency and blockchain risk with system and organization controls (soc) attestation, 2023.

[203] S. K. Lo, X. Xu, M. Staples, and L. Yao. Reliability analysis for blockchain oracles. *Computers & Electrical Engineering*, 83:106582, 2020.

[204] LongForWisdom. Flash loans and securing the maker protocol. *URl: https://forum.makerdao.com/t/urgent-flash-loans-and-securing-the-maker-protocol/4901*, 2020.

[205] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 254–269. ACM, 2016.

[206] S. Madnick and H. Zhu. Improving data quality through effective use of data semantics. *Data & Knowledge Engineering*, 59(2):460–475, 2006.

[207] MakerDAO. Introducing oracles v2 and defi feeds, 2019.

[208] K. Malinova and A. Park. Market design with blockchain technology. 2017.

[209] D. Maram, H. Malvai, F. Zhang, N. Jean-Louis, A. Frolov, T. Kell, T. Lobban, C. Moy, A. Juels, and A. Miller. Candid: can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1348–1366. IEEE, 2021.

[210] Y. Marcus, E. Heilman, and S. Goldberg. Low-resource eclipse attacks on ethereum's peer-to-peer network. Cryptology ePrint Archive, Report 2018/236, 2018.

[211] J. W. Markham. Front-running-insider trading under the commodity exchange act. *Cath. UL Rev.*, 38:69, 1988.

[212] F. Massacci, C. N. Ngo, J. Nie, D. Venturi, and J. Williams. Futuresmex: secure, distributed futures market exchange. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 335–353. IEEE, 2018.

[213] G. Maxwell. Confidential transactions. *URL: https://people. xiph. org/˜ greg/-confidential_values. txt (Accessed 09/05/2016)*, 2015.

[214] P. McCorry. Enablement of mev and the morals of extracting, 2023.

[215] P. McCorry, A. Hicks, and S. Meiklejohn. Smart contracts for bribing miners. IACR Cryptology ePrint Archive, 2018.

[216] P. McCorry, S. F. Shahandashti, and F. Hao. A smart contract for boardroom voting with maximum voter privacy. In *International Conference on Financial Cryptography and Data Security*, pages 357–375. Springer, 2017.

[217] E. Medvedev. Python scripts for etl (extract, transform and load) jobs for ethereum blocks. `https://github.com/medvedev1088/ethereum-etl`, 2018.

[218] S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*, pages 120–130. IEEE, 1999.

[219] T. Micro. Malvertising campaign abuses google's doubleclick to deliver cryptocurrency miners. `https://blog.trendmicro.com/trendlabs-security-intelligence/malvertising-campaign-abuses-googles-doubleclick-to-deliver-cryptocurrency-miners/`, 2018. Accessed: 2018-01-31.

[220] I. Miers, C. Garman, M. Green, and A. D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 397–411. IEEE, 2013.

[221] A. Miller, M. Moeser, K. Lee, and A. Narayanan. An Empirical Analysis of Linkability in the Monero Blockchain. Technical report, arXiv, 2017.

[222] A. Miller, M. Möser, K. Lee, and A. Narayanan. An empirical analysis of linkability in the monero blockchain. *arXiv preprint*, 1704, 2017.

[223] A. Moin, K. Sekniqi, and E. G. Sirer. Sok: A Classification framework for stablecoin designs. In *Financial Cryptography*, 2020.

[224] P. Momeni, S. Gorbunov, and B. Zhang. Fairblock: Preventing blockchain front-running with minimal overheads. In *International Conference on Security and Privacy in Communication Systems*, pages 250–271. Springer, 2022.

[225] S. Mondal, K. P. Wijewardena, S. Karuppuswami, N. Kriti, D. Kumar, and P. Chahal. Blockchain inspired rfid-based information architecture for food supply chain. *IEEE Internet of Things Journal*, 6(3):5803–5813, 2019.

[226] Monero. MONERO private digital currency. `https://getmonero.org/`, 2014. Accessed: 2017-11-20.

[227] J. H. Moor. What is computer ethics? *Metaphilosophy*, 16(4):266–275, 1985.

[228] S. Moosavi and J. Clark. Ghazal: toward truly authoritative web certificates using ethereum. In *Financial Cryptography and Data Security: FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers 22*, pages 352–366. Springer, 2019.

[229] A. Müller and M. Grandi. Weather derivatives: a risk management tool for weather-sensitive industries. *The Geneva Papers on Risk and Insurance. Issues and Practice*, 25(2):273–287, 2000.

[230] T. Mursch. Cryptojacking malware coinhive found on 30,000+ websites. `https://badpackets.net/cryptojacking-malware-coinhive-found-on-30000-websites/`, 2017. Accessed: 2018-01-20.

[231] M. Musch, C. Wressnegger, M. Johns, and K. Rieck. Web-based cryptojacking in the wild. *arXiv preprint arXiv:1808.09474*, 2018.

[232] M. Musch, C. Wressnegger, M. Johns, and K. Rieck. Thieves in the browser: Web-based cryptojacking in the wild. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–10, 2019.

[233] N. Mutual. Responsible vulnerability disclosure. *URl: https://medium.com/nexus-mutual/responsible-vulnerability-disclosure-ece3fe3bcefa*, 2020.

[234] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[235] NakedSecurity. Unsecured aws led to cryptojacking attack on la times. `https://nakedsecurity.sophos.com/2018/02/27/unsecured-aws-led-to-cryptojacking-attack-on-la-times/`, 2018. Accessed: 2018-02-28.

[236] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016.

[237] A. Narayanan and J. Clark. Bitcoin's academic pedigree. *Communications of the ACM*, 60(12):36–45, 2017.

[238] F. N. Naseem, A. Aris, L. Babun, E. Tekiner, and A. S. Uluagac. Minos: A lightweight real-time cryptojacking detection system. In *NDSS*, 2021.

[239] NEST. Nest protocol: A distributed price oracle network. *URl: https://nestprotocol.org/doc/ennestwhitepaper.pdf*, 2020.

[240] T. Neudecker, P. Andelfinger, and H. Hartenstein. A simulation model for analysis of attacks on the bitcoin peer-to-peer network. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1327–1332. IEEE, 2015.

[241] T. Neudecker and H. Hartenstein. Short paper: An empirical analysis of blockchain forks in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 84–92. Springer, 2019.

[242] R. Ning, C. Wang, C. Xin, J. Li, L. Zhu, and H. Wu. Capjack: Capture in-browser crypto-jacking by deep capsule network through behavioral analysis. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1873–1881. IEEE, 2019.

[243] S. Noether. Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098, 2015. `https://eprint.iacr.org/2015/1098`.

[244] N. J. D. of Consumer Affairs. New jersey division of consumer affairs obtains settlement with developer of bitcoin-mining software found to have accessed new jersey computers without users knowledge or consent. http://nj.gov/oag/newsreleases15/pr20150526b.html, 2015. Accessed: 2018-01-20.

[245] B. of England. Financial stability in focus: Cryptoassets and decentralised finance. *Bank of England*, 2022.

[246] U. D. C. S. D. of Florida. Deposition of gavin a. andresen, david kleiman, w&k info defense research, llc, plaintiffs, v. craig wright, defendant. 2020.

[247] O. of the Superintendent of Financial Institutions. Interim arrangements for the regulatory capital and liquidity treatment of cryptoasset exposures, 2022.

[248] Opera. New year, new browser. opera 50 introduces anti-bitcoin mining tool. http://blogs.opera.com/desktop/2018/01/opera-50-introduces-anti-bitcoin-mining-tool/, 2018. Accessed: 2018-01-20.

[249] Optimism. Optimism is ethereum, scaled. online, 2022.

[250] C. Oracle. Explicit staking in chainlink 2.0: An overview. *URl: https://blog.chain.link/explicit-staking-in-chainlink-2-0/*, 2021.

[251] P. Papadopoulos, P. Ilia, and E. P. Markatos. Truth in web mining: Measuring the profitability and cost of cryptominers as a web monetization model. *arXiv preprint arXiv:1806.01994*, 2018.

[252] M. Paquet-Clouston, B. Haslhofer, and B. Dupont. Ransomware payments in the bitcoin ecosystem. *Journal of Cybersecurity*, 5(1):tyz003, 2019.

[253] C. F. Paul Brody. Ey blockchain analyzer: Reconciler. *Ernst & Young*, 2022.

[254] U. Pavloff, Y. Amoussou-Guenou, and S. Tucci-Piergiovanni. Ethereum proof-of-stake under scrutiny. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, pages 212–221, 2023.

[255] PeckShield. bzx hack ii full disclosure (with detailed profit analysis), 2020.

[256] I. G. A. Pernice, S. Henningsen, R. Proskalovich, M. Florian, and H. Elendner. Monetary stabilization in cryptocurrencies: Design approaches and open questions. In *CVCBT*, 2019.

[257] J. Peterson and J. Krug. Augur: a decentralized, open-source platform for prediction markets. *arXiv preprint arXiv: 1501.01042*, 2015.

[258] I. Petrov, L. Invernizzi, and E. Bursztein. Coinpolice: Detecting hidden cryptojacking attacks with neural networks. *arXiv preprint arXiv:2006.10861*, 2020.

[259] C. Petty. A look at the status.im ico token distribution. `https://medium.com/the-bitcoin-podcast-blog/a-look-at-the-status-im-ico-token-distribution-f5bcf7f00907`, 2017. Accessed: 2018-06-10.

[260] C. Pierrot and B. Wesolowski. Malleability of the blockchain's entropy. *Cryptography and Communications*, 10(1):211–233, 2018.

[261] E. Pimentel and E. Boulianne. Blockchain in accounting research and practice: Current trends and future opportunities. *Accounting Perspectives*, 19(4):325–361, 2020.

[262] E. Pimentel, E. Boulianne, S. Eskandari, and J. Clark. Systemizing the challenges of auditing blockchain-based assets. *Journal of Information Systems*, 35(2):61–75, 2021.

[263] E. Piqueras. Generalized ethereum frontrunners, an implementation and a cheat. 2019.

[264] PoolTogether. Pooltogether is a no-loss, audited savings game powered by blockchain technology. *URl: https://www.pooltogether.com/*, 2020.

[265] A. Posadzki and D. Milstead. Crypto companies struggle to find auditors amid heightened scrutiny. *The Globe and Mail (March 22) B*, 1, 2019.

[266] B. Protocol. Bandchain whitepaper. *URl: https://docs.bandchain.org/whitepaper/system-overview.html*, 2020.

[267] K. Qin, L. Zhou, and A. Gervais. Quantifying blockchain extractable value: How dark is the forest? In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 198–214. IEEE, 2022.

[268] K. Qin, L. Zhou, B. Livshits, and A. Gervais. Attacking the defi ecosystem with flash loans for fun and profit. *arXiv preprint arXiv:2003.03810*, 2020.

[269] K. Qin, L. Zhou, B. Livshits, and A. Gervais. Attacking the defi ecosystem with flash loans for fun and profit. In *International conference on financial cryptography and data security*, pages 3–32. Springer, 2021.

[270] H. Qureshi. Introducing cofix: a next-generation amm. *URl: https://medium.com/dragonfly-research/introducing-cofix-a-next-generation-amm-199aea686b6b*, 2020.

[271] R. Radner and A. Schotter. The sealed-bid mechanism: An experimental study. *Journal of Economic Theory*, 48(1):179–220, 1989.

[272] H. Ragab, A. Milburn, K. Razavi, H. Bos, and C. Giuffrida. Crosstalk: Speculative data leaks across cores are real. In *IEEE Symposium on Security and Privacy*, 2021.

[273] R. Rahimian. Multiple withdrawal attack. 2018.

[274] R. Rahimian, S. Eskandari, and J. Clark. Resolving the multiple withdrawal attack on erc20 tokens. In *2019 IEEE European symposium on security and privacy workshops (EuroS&PW)*, pages 320–329. IEEE, 2019.

[275] E. RANDAO. Block proposal. `https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/block-proposal/`, 2023. Accessed: 2023-12-18.

[276] C. Reitwiessner. An update on integrating zcash on ethereum (zoe). *URL: https://blog.ethereum.org/2017/01/19/update-integrating-zcash-ethereum/*, 2017.

[277] J. D. P. Rodriguez and J. Posegga. Rapid: Resource and api-based detection against in-browser miners. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 313–326, 2018.

[278] M. Rosenfeld. Analysis of bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.

[279] N. Rückeshäuser. Do we really want blockchain-based accounting? decentralized consensus as enabler of management override of internal controls. 2017.

[280] S. Ruoti, B. Kaiser, A. Yerukhimovich, J. Clark, and R. Cunningham. Blockchain technology: What is it good for? *Communications of the ACM*, 63(1):46–53, 2020.

[281] J. Rüth, T. Zimmermann, K. Wolsing, and O. Hohlfeld. Digging into browser-based crypto mining. In *Proceedings of the Internet Measurement Conference 2018*, pages 70–76, 2018.

[282] S. Ruwhof. Massive child porn site is hiding in plain sight, and the owners behind it. `https://sijmen.ruwhof.net/weblog/1782-massive-child-porn-site-is-hiding-in-plain-sight-and-the-owners-behind-it`, 2017. Accessed: 2018-01-20.

[283] G. Rydstedt, E. Bursztein, D. Boneh, and C. Jackson. Busting frame busting: a study of clickjacking vulnerabilities at popular sites. In *IEEE SSP*, volume 2, 2010.

[284] M. Saad, A. Khormali, and A. Mohaisen. End-to-end analysis of in-browser cryptojacking. *arXiv preprint arXiv:1809.02152*, 2018.

[285] M. Salehi, J. Clark, and M. Mannan. Red-black coins: Dai without liquidations. In *Financial Cryptography: DeFi*, 2021.

[286] S. Sallam and B. D. Beheshti. A survey on lightweight cryptographic algorithms. In *TENCON 2018-2018 IEEE Region 10 Conference*, pages 1784–1789. IEEE, 2018.

[287] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy (SP)*, pages 459–474. IEEE, 2014.

[288] Securities, E. Commission, et al. Report of the special study of the options markets. *Chapter I, Introduction, Figure*, 5, 1979.

[289] Securities and E. C. (SEC). Financial reporting manual: Topic 4—independent accountants' involvement. 2009.

[290] A. Shevchenko. Uniswap and automated market makers, explained. *URl: https://cointelegraph.com/explained/uniswap-and-automated-market-makers-explained*, 2020.

[291] D. Siegel. Understanding the dao attack. *URL http://www.coindesk.com/understanding-dao-hack-journalists*, 2016.

[292] C. Signer. Gas cost analysis for ethereum smart contracts. Master's thesis, ETH Zurich, Department of Computer Science, 2018.

[293] D. Sillars. The performance impact of cryptocurrency mining on the web. `https://discuss.httparchive.org/t/the-performance-impact-of-cryptocurrency-mining-on-the-web/1126`, 2017. Accessed: 2017-12-20.

[294] O. Solmaz. The anatomy of a block stuffing attack. *https://osolmaz.com/2018/10/18/anatomy-block-stuffing/*, 2018.

[295] K. Sonar and H. Upadhyay. A survey: Ddos attack on internet of things. *International Journal of Engineering Research and Development*, 10(11):58–63, 2014.

[296] A. Sotirakopoulos, K. Hawkey, and K. Beznosov. On the challenges in usable security lab studies: Lessons learned from replicating a study on SSL warnings. In *SOUPS*, 2011.

[297] M. W. Stanislaw Drozdz, Jaroslaw Kwapien. What is mature and what is still emerging in the cryptocurrency market? *Entropy*, 25(5):772, 2023.

[298] C. Stathakopoulou, S. Rüsch, M. Brandenburger, and M. Vukolić. Adding fairness to order: Preventing front-running attacks in bft protocols using tees. In *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*, pages 34–45. IEEE, 2021.

[299] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor. Crying wolf: An empirical study of SSL warning effectiveness. In *USENIX Security*, 2009.

[300] Synthetix. Synthetix response to oracle incident. *URl: https://blog.synthetix.io/response-to-oracle-incident/*, 2019.

[301] Synthetix. A derivatives liquidity protocol. *URl: https://www.synthetix.io/*, 2020.

[302] P. Sztorc. Truthcoin. *peer-to-peer oracle system and prediction marketplace.*, 2015.

[303] R. Tahir, M. Huzaifa, A. Das, M. Ahmad, C. Gunter, F. Zaffar, M. Caesar, and N. Borisov. Mining on someone else's dime: Mitigating covert mining operations in clouds and enterprises. In *RAID*, 2017.

[304] J. Z. Tan, T. Merk, S. Hubbard, E. R. Oak, J. Pirovich, E. Rennie, R. Hoefer, M. Zargham, J. Potts, C. Berg, et al. Open problems in daos. *arXiv preprint arXiv:2310.19201*, 2023.

[305] D. Tanana. Behavior-based detection of cryptojacking malware. In *2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, pages 0543–0545. IEEE, 2020.

[306] C. team. Cryptokitties, 2018. Accessed: 2018-08-31.

[307] S. Team. The status network, a strategy towards mass adoption of ethereum, 2017. Accessed: 2018-06-10.

[308] E. Tekiner, A. Acar, A. S. Uluagac, E. Kirda, and A. A. Selcuk. Sok: cryptojacking malware. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 120–139. IEEE, 2021.

[309] Tellor. A decentralized Oracle.

[310] Tellor-io. Eip-2362: Pull oracle interface. *URl: https://github.com/tellor-io/EIP-2362*, 2020.

[311] TheGuardian. Ads dont work so websites are using your electricity to pay the bills. `https://www.theguardian.com/technology/2017/sep/27/pirate-bay-showtime-ads-websites-electricity-pay-bills-cryptocurrency-bitcoin`, 2017. Accessed: 2017-11-20.

[312] ThePirateBay. The galaxys most resilient bittorrent site. `https://thepiratebay.org/blog/242`, 2017. Accessed: 2017-11-20.

[313] TheRegister. Cbs's showtime caught mining crypto-coins in viewers' web browsers. https://www.theregister.co.uk/2017/09/25/showtime_hit_with_coinmining_script/, 2017. Accessed: 2018-01-20.

[314] TheRegister. Crypto-jackers enlist google tag manager to smuggle alt-coin miners. https://www.theregister.co.uk/2017/11/22/cryptojackers_google_tag_manager_coin_hive/, 2017. Accessed: 2018-01-20.

[315] TheRegister. Lets get ready to grumble! ufc secretly choke slams browsers with monero miners. https://www.theregister.co.uk/2017/11/07/ufc_coin_hive/, 2017.

[316] TheRegister. Uk ico, uscourts.gov... thousands of websites hijacked by hidden crypto-mining code after popular plugin pwned. https://www.theregister.co.uk/2018/02/11/browsealoud_compromised_coinhive/, 2018. Accessed: 2018-02-28.

[317] TheVerge. Hotel caught injecting advertising into webpages on complimentary wi-fi network. https://www.theverge.com/2012/4/7/2931600/hotel-caught-injecting-advertising-into-web-pages-on-complimentary-wi, 2012. Accessed: 2017-12-08.

[318] TheVerge. Showtime websites secretly mined user cpu for cryptocurrency. https://www.theverge.com/2017/9/26/16367620/showtime-cpu-cryptocurrency-monero-coinhive, 2017. Accessed: 2017-11-20.

[319] F. Tian. An agri-food supply chain traceability system for china based on rfid & blockchain technology. In *2016 13th international conference on service systems and service management (ICSSSM)*, pages 1–6. IEEE, 2016.

[320] TLSnotary. A mechanism for independently audited https sessions. *URl: https://tlsnotary.org/TLSNotary.pdf*, 2014.

[321] C. F. Torres, R. Camino, et al. Frontrunner jones and the raiders of the dark forest: An empirical study of frontrunning on the ethereum blockchain. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1343–1359, 2021.

[322] H. Treasury. Future financial services regulatory regime for crypto-assets. *Consultation and call for evidence*, 2023.

[323] UMA. Uma data verification mechanism: Adding economic guarantees to blockchain oracles. *URl: https://github.com/UMAprotocol/whitepaper/blob/master/UMA-DVM-oracle-whitepaper.pdf*, 2020.

[324] Uniswap. Uniswap oracle. *URl: https://uniswap.org/docs/v2/core-concepts/oracles*, 2020.

[325] N. Van der Meulen. Diginotar: Dissecting the first dutch digital disaster. *Journal of strategic security*, 6(2):46–58, 2013.

[326] N. van Saberhagen. Cryptonote v 2. 0. https://bytecoin.org/old/whitepaper.pdf, 2013.

[327] S. Varlioglu, B. Gonen, M. Ozer, and M. Bastug. Is cryptojacking dead after coinhive shutdown? In *2020 3rd International Conference on Information and Computer Technologies (ICICT)*, pages 385–389. IEEE, 2020.

[328] M. Vasek, M. Thornton, and T. Moore. Empirical analysis of denial-of-service attacks in the bitcoin ecosystem. In *International conference on financial cryptography and data security*, pages 57–71. Springer, 2014.

[329] R. Ver and J. Wu. Bitcoin cash planned network upgrade is complete, 2018.

[330] J. Vermorel, A. Séchet, S. Chancellor, and T. van der Wansem. Canonical transaction ordering for bitcoin. 2018. Accessed: 2018-12-07.

[331] F. Vogelsteller and V. Buterin. Erc-20 token standard. https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md, 2015. Accessed: 2018-08-31.

[332] A. Wahrstätter, J. Ernstberger, A. Yaish, L. Zhou, K. Qin, T. Tsuchiya, S. Steinhorst, D. Svetinovic, N. Christin, M. Barczentewicz, et al. Blockchain censorship. *arXiv preprint arXiv:2305.18545*, 2023.

[333] WallStreetJournal. Your computer may be making bitcoin for hackers. https://www.wsj.com/articles/hackers-latest-move-using-your-computer-to-mine-bitcoin-1509102002, 2017. Accessed: 2018-01-20.

[334] T. Walther. Multi-token batch auctions with uniform clearing prices. 2018.

[335] Y. Wang. Automated market makers for decentralized finance (defi). *arXiv preprint arXiv:2009.01676*, 2020.

[336] Y. Wang, Q. Zhang, K. Li, Y. Tang, J. Chen, X. Luo, and T. Chen. ibatch: saving ethereum fees via secure and cost-effective batching of smart-contract invocations. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 566–577, 2021.

[337] W. Warren. Front-running, griefing and the perils of virtual settlement. https://blog.0xproject.com/front-running-griefing-and-the-perils-of-virtual-settlement-part-1-8554ab283e97, 2017. (Accessed on 08/14/2018).

[338] W. Warren and A. Bandeali. 0x: An open protocol for decentralized exchange on the ethereum blockchain. *URl: https://github. com/0xProject/whitepaper*, 2017.

[339] Washingtonpost. Hackers have turned politifact's website into a trap for your pc. `https://www.washingtonpost.com/news/the-switch/wp/2017/10/13/hackers-have-turned-politifacts-website-into-a-trap-for-your-pc`, 2017. Accessed: 2018-01-20.

[340] N. Webb. A fork in the blockchain: income tax and the bitcoin/bitcoin cash hard fork. *North Carolina Journal of Law & Technology*, 19(4):283, 2018.

[341] E. Website. Proposer-builder separation. `https://ethereum.org/nl/roadmap/pbs/`, 2023. Accessed: 2023-12-18.

[342] B. Weintraub, C. F. Torres, C. Nita-Rotaru, and R. State. A flash (bot) in the pan: measuring maximal extractable value in private pools. In *Proceedings of the 22nd ACM Internet Measurement Conference*, pages 458–471, 2022.

[343] whiterabbit. Black thursday for makerdao: $8.32 million was liquidated for 0 dai, 2020.

[344] V. Wikipedia. Important milestones of the bitcoin project. `https://en.bitcoin.it/wiki/Category:History`, 2009. Accessed: 2018-01-18.

[345] D. Z. J. Williamson. The aztec protocol. *URL: https://github.com/AztecProtocol/AZTEC/*, 2018.

[346] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. *computer*, 35(10):54–62, 2002.

[347] G. Wood et al. Ethereum: A secure decentralised generalised transaction ledger. Technical Report 2014, Ethereum, 2014.

[348] M. F. Worboys and S. M. Deen. Semantic heterogeneity in distributed geographic databases. *ACM Sigmod Record*, 20(4):30–34, 1991.

[349] J. Wyke. The zeroaccess botnet: Mining and fraud for massive financial gain. *Sophos Technical Paper*, 2012.

[350] S. Yang, F. Zhang, K. Huang, X. Chen, Y. Yang, and F. Zhu. Sok: Mev countermeasures: Theory and practice. *arXiv preprint arXiv:2212.05111*, 2022.

[351] A. Yayla, E. Dincelli, and S. Parameswaran. A mining town in a digital land: Browser-based cryptocurrency mining as an alternative to online advertising. *Information Systems Frontiers*, pages 1–23, 2023.

[352] T. Yu, Z. Lin, and Q. Tang. Blockchain: The introduction and its application in financial accounting. *Journal of Corporate Accounting & Finance*, 29(4):37–47, 2018.

[353] Zaugust. Cofix: A computable trading system. *URl: https://cofix.io/doc/Trading_Compensation_CoFiX.pdf*, 2020.

[354] P. J. Zelbst, K. W. Green, V. E. Sower, and P. L. Bond. The impact of rfid, iiot, and blockchain technologies on supply chain transparency. *Journal of Manufacturing Technology Management*, 2019.

[355] D. A. Zetzsche, R. P. Buckley, D. W. Arner, and L. Föhr. The ico gold rush: It's a scam, it's a bubble, it's a super challenge for regulators. 2018.

[356] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi. Town crier: An authenticated data feed for smart contracts. In *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security*, pages 270–282. ACM, 2016.

[357] F. Zhang, S. K. D. Maram, H. Malvai, S. Goldfeder, and A. Juels. Deco: Liberating web data using decentralized oracles for tls. *arXiv preprint arXiv:1909.00938*, 2019.

[358] H. Zhang, L.-H. Merino, V. Estrada-Galinanes, and B. Ford. F3b: A low-latency commit-and-reveal architecture to mitigate blockchain front-running. *arXiv preprint arXiv:2205.08529*, 2022.

[359] S. Zhang and J.-H. Lee. Double-spending with a sybil attack in the bitcoin decentralized network. *IEEE Transactions on Industrial Informatics*, 15(10):5715–5722, 2019.

[360] Z. Zhang, Z. Lin, M. Morales, X. Zhang, and K. Zhang. Your exploit is mine: Instantly synthesizing counterattack smart contract. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 1757–1774, 2023.

[361] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, and Y.-a. Tan. Secure multi-party computation: theory, practice and applications. *Information Sciences*, 476:357–372, 2019.

[362] L. Zhou, K. Qin, and A. Gervais. A2mm: Mitigating frontrunning, transaction reordering and consensus instability in decentralized exchanges. *arXiv preprint arXiv:2106.07371*, 2021.

[363] H. Zhu. Do dark pools harm price discovery? *The Review of Financial Studies*, 27(3):747–789, 2014.