Politecnico di Torino

Master's degree in Computer Engineering Cybersecurity
Graduation Session April 2023

Master's Thesis

Analysis of blockchain security and possible solutions to the oracle problem

Supervisor:
Gianpiero Cabodi

Candidate:
Davide Mammone

A.a. 2020/2021

Indice

Introduction

Blockchain technology has rised as a revolutionary development in the digital world, offering secure and efficient solutions for various industries. The decentralized nature of blockchain, along with its immutable and distributed ledger, has made it an attractive solution for numerous applications. One of the key components of blockchain technology is oracles, which provide external data to smart contracts on the blockchain. However, oracles also pose significant security risks, introducing a potential point of failure or manipulation.

To address these security concerns, this thesis aims to provide a comprehensive understanding of oracles in blockchain technology, examining their concepts, applications and security issues. The thesis is divided into four chapters, each exploring a different aspect of oracles and their role in blockchain technology.

The first chapter focuses on the fundamentals of blockchain technology, introducing the concept of a decentralized, immutable and distributed digital ledger that maintains a continuously growing list of transactions. The chapter also explores the concept of consensus, in the form of Proof of Work and Proof of Stake, which is essential for maintaining the integrity of the blockchain. Different types of blockchain are introduced, including public and private, along with their respective advantages and disadvantages.

This chapter examines smart contracts, which are self-executing agreements with terms written directly into code, and their benefits, such as efficiency, automation and transparency.

The chapter also delves into the concept of oracles, which are third-party services that provide external data to a blockchain. The chapter also provides examples of possible use cases for oracles, such as in the areas of insurance, supply chain management and finance.

The second chapter explores the security issues associated with oracles, highlighting their importance in maintaining the integrity of blockchain systems. The chapter provides a comprehensive overview of the different types of oracle attacks, discussing in detail how they work and their impacts on the blockchain ecosystem. To address these security risks, the chapter discusses several possible solutions, including the use of multiple oracles, trusted execution environments, cryptographic proofs and reputation-based systems.

The third chapter of the thesis provides a detailed comparison between two popular oracle networks, Chainlink and Provable, highlighting their differences in terms of security, functionality, design choices and costs. We then analyse the source code used in each network, examining the design choices and implementation details that contribute to their security and performance.

In the fourth chapter, we examine two different approaches to address the oracle problem in the blockchain space: L2 Zk Rollups and Optimistic Oracles. L2 Zk Rollups use zero-knowledge proofs and off-chain aggregation to improve scalability and privacy while maintaining security. This is achieved by aggregating and processing multiple transactions off-chain and then submitting a single proof to the blockchain. This approach allows for faster and cheaper processing of transactions, but at the expense of increased complexity and reduced transparency. On the other hand, Optimistic Oracles use a consensus mechanism among a group of validators to resolve disputes over the validity of data inputs. This approach is more straightforward, but it relies on the honesty of the validators and may require more time to resolve disputes.

Overall, this thesis provides a comprehensive analysis of the challenges and solutions related to the use of oracles in blockchain technology. By examining the fundamental concepts of blockchain technology and exploring the various types of oracles, this thesis aims to provide a solid foundation for understanding the importance of oracles in the blockchain ecosystem.

Chapter 1

Web3

Oracles are the means by which smart contracts can express their full potential within the blockchain; without them, one could not speak of decentralised finance. In the eyes of a reader who has never delved into the world of crypto, these terms might not say anything, so this chapter will introduce everything related to Web 3.0.

1.1. Blockchain

1.1.1. History of blockchain

The concept of blockchain is not a recent thing, in fact it dates back to the 1980s and 1990s, where a couple of cryptography researchers (Stuart Haber & Wakefield Scott Stornetta) introduced a solution for time-stamping digital documents, so that they could not be backdated or altered. Their patent, however, expired in 2004 due to the unsuccessful use of this technology, which would be taken up in 2008 by a person or group known by the identification of Satoshi Nakamoto. Releasing and publishing the Bitcoin white-paper in 2008 he establishes a new era of coin. [1]

1.1.2. What is blockchain

A blockchain is a highly secure, reliable, and decentralized network that allows people to record transaction activity, store data, and exchange value in a distributed ledger that is not controlled by any central authority, but instead maintained by computers all around the world.[1]

[1] Chainlink, https://blog.chain.link/what-is-blockchain/, last update January 24, 2022, last access October 13, 2022

Blockchain is the technology on which the whole cryptocurrency or web3 system is based, it allows information, data or transactions to be exchanged, without having to rely on third parties in a secure and trustless manner (i.e., there is no need to trust someone else while performing a transaction).

If one uses a traditional transaction as an illustration, it will become clear that there are more than two parties involved in the exchange of money and that both of them must rely on a central organization - in this case a bank - to which the money must be transferred before the same bank can then distribute it to the recipient. With the blockchain, you can fully cut out the intermediary without sacrificing the security that he or she offers a transaction.

The fundamental characteristic of blockchain technology is security, due to the transparency and the fact that any individual is able to verify any information contained in the ledger. Furthermore, thanks to the way the chain is implemented, every piece of information written on it results immutable.

1.1.3. Blockchain immutability

The blockchain, as the name says, is formed by a chain of blocks, within which various information such as pending transactions are stored, to be processed and validated by each node constituting the network. The fact that each node must verify and validate every transaction, makes an attack against it virtually impossible. Each transaction is made secure through a hashing process, and each node's hash contains metadata about the hash of the previous block, making any attempt to modify a block futile. Note that immutability does not have only positive meanings. There are many possibilities for incorrect information, or for sensitive data that should not have been disclosed to be entered into the chain. In a normal list certain information could be changed or removed altogether, given its lack of connection to the outside world the problem would have been solved; but given the immutability of the blockchain this is not possible, everything written on it - whether favourable or unfavourable - will remain written on the chain.

1.1.4. Consensus

Consensus is the process put in place by the blockchain nodes, whereby each transaction is deemed valid or not. This agreement is achievable by the usage of different mechanisms, the most used are proof of work (PoW) and proof of stake (PoS).

1.1.4.1. Proof of Work (PoW)

Proof of work is the process whereby miners[2] compete with each other to validate the various blocks of the chain by performing a calculation to solve a high-level cryptographic problem, which requires considerable computing power. The rewards given to miners who succeed in solving the computation may differ; in the Bitcoin blockchain, for example, one is rewarded with Bitcoins. [2] Although this mechanism is considered the most secure and reliable, the great disadvantage of this solution is the high energy consumption required to complete the operation; in fact, solutions involving the exclusive use of entire server rooms in order to mine bitcoins are not uncommon. The downside is the total disregard for the environment, given the substantial number of pollutants released into the atmosphere due to the intensive use and consumption of electricity.[3]

The fundamental tenet of Proof-of-Work (PoW) is that a node is randomly chosen based on how much effort it expends, or rather, how much computer power it requires to solve difficult mathematical problems called hash-puzzles. The node must locate a nonce (literally, "number once used") and build a hash of the block by double hashing in order to create a block. Simply put, this number will be the input for a hash function along with other elements of a block header. For a block to be accepted, the value given by the hash function's output cannot be greater than

[2] Owners of the computers/super computers that perform the calculation.
[3] The well-known Elon Musk, CEO of Tesla, after purchasing bitcoins for a total of $1.5 billion and announcing the possibility of making purchases on the site through the aforementioned tokens, retraces his steps by declaring: "we are concerned about the rapid increase in the use of fossil fuels for mining and bitcoin transactions, in particular coal, which has the worst emissions of any fuel".

a specific level of difficulty. This can be summed up by the inequality shown below:

H(nonce || prev_hash || tx || tx || … || tx) < target

The number of initial and subsequent zeros in the output hash code must be at least as many as the specified level of difficulty. The block header's nBit field serves as a representation of the latter. As a reference, the acceptable hash for node validation should resemble the following if the target is set to 8:

000000006c8b2dc13c7b6dc990b6148ae6de16b4d8ffa72d08493d9656d126fb64f5acb1

The only header parameter that may be changed is the nonce, which is typically initialized to 0. The nonce is increased and the procedure is repeated until a suitable hash value is found each time the acquired block's hash deviates from the intended value. The only logical method to achieve this is by multiple random attempts, hoping to get fortunate before the others in the absence of any strategy or computation to discover a legitimate nonce.

To demonstrate to the other nodes that its work is accurate, a node that discovers a solution sends the "result" to them (proof-of-work). The block is now formally validated when the other nodes confirm that the inequality specified above is satisfied. The block is subsequently added to the blockchain, whereupon the node who discovered the solution is rewarded with fresh bitcoins and all associated transaction fees.

1.1.4.2. Proof of Stake (PoS)

It is a new alternative method to PoW, where there is no more need for miners and high performing computers. The reason for this is that consensus is reached by a process known as staking; the new validators must give their coins as collateral in order to win a sort of bet on the valid block – as a matter of fact, the more coins you collateralize, the more probabilities you have to be the one chosen for validating the block. All that glitters is not gold though; in fact, PoS comes with a major drawback, which is its slowness. There is the possibility that an

excessive number of validators will slow down the network as their number is directly proportional to the time it takes to reach consensus. The development and implementation of this mechanism will be more complex due to its complicated algorithms, and furthermore, the system favours persons with more tokens as these will always be more likely to be chosen as validators.

1.1.5. Blockchain types

Distributed ledgers, on which blockchains are built, have been used for managing data at the corporate level for many years. But it wasn't until lately that they started to gain popularity and attention due of cryptocurrencies, which popularized the idea.

Depending on how the blockchain is set up, it is possible to regulate the data contained on its blocks as well as the actions taken by its many users. Blockchains often provide certain functions, and users may access or perform a variety of actions.

Public blockchains are accessible to everyone, private blockchains are only accessible to a small number of users, and permissioned blockchains are a mix of both public and private blockchains that anybody may access provided they have the administrators' permission.

1.1.5.1. Public blockchain

A public blockchain is one that allows anybody to join and take part in the essential operations of the blockchain network. It is possible for anybody to view, publish, and audit the current activity on a public blockchain network, which contributes to the self-governed, decentralized aspect that is frequently highlighted when discussing blockchain technology.

A public network runs on an incentive system that motivates new users to sign up and maintain the network's flexibility. From the perspective of an operation that is really decentralized, democratized, and authority-free, public blockchains present a particularly attractive answer. The fact that public blockchains can act as the

foundation for almost any decentralized solution makes them incredibly important. Additionally, a secured public blockchain is protected from data breaches, hacking attempts, and other cybersecurity problems by the large number of network users that join it. A blockchain is safer the more participants it has.

On the other hand, anyone can observe transaction amounts and the addresses involved on public blockchains. The user's anonymity is compromised if the address owners are made public. Public blockchains can draw users with motivations that might not be honest, as the majority of them are made for cryptocurrencies, which are a prime target for thieves and hackers due to their inherent worth.

1.1.5.2. Private blockchain

Participants can join a private blockchain network only through an invitation where their identity or other required information is authentic and verified. The validation is carried out either by the network operator(s) or by the network itself, using smart contracts or other automated approval techniques, to carry out a well-defined set protocol.

Private blockchains have restrictions on who can use the network. The network's private nature can limit which users can run the consensus process that determines the mining rights and rewards if it has mining capabilities. The shared ledger might also be maintained by a small group of users. The owner or operator has the authority to alter, amend, or remove any required blockchain entries as necessary or appropriate.

A private blockchain is not decentralized. It is a distributed ledger[4] that functions as a closed database protected by cryptographic principles and the requirements of

[4] A distributed ledger is a database that is widely accessible and cooperatively shared across numerous locations, organizations, or geographies. It enables transactions to have "witnesses" in the public eye. Each participant can access and keep a duplicate copy of any recordings that are shared among network nodes. Any additions or modifications to the ledger are immediately reflected and duplicated to all participants. It contrasts with centralized ledgers, which is the kind of ledger that most businesses utilize, and that provide a single point of failure, making it more vulnerable to fraud and cyberattacks.

the organization. Nobody without authorization is allowed to operate a full node, conduct transactions, or validate or authenticate blockchain modifications.

Private blockchains put more emphasis on efficiency and immutability while lessening the emphasis on preserving user privacy and fostering transparency. These are crucial elements in many different enterprise and commercial fields, including supply, logistics, payroll, finances, and accounting.

1.1.5.3. Permissioned blockchain

Permissioned blockchains are a customizable mix between the public and private ones.

The benefit of a permissioned blockchain is that anyone can join the network following a proper identity verification procedure. Some grant exclusive permissions to carry out only particular actions on a network. This enables users to carry out specific tasks like reading, accessing, or entering data on the blockchain.

Many activities are possible with permissioned blockchains, but Blockchain-as-a-Service (BaaS)—a blockchain built to scale for the requirements of many businesses or jobs that the providers rent out to other businesses—is the one that interests businesses the most.

Depending on how they are set up, permissioned blockchains can have the same drawbacks as public and private blockchains. Permissioned blockchains have a number of drawbacks, including the fact that they are susceptible to hacking because they need internet connectivity. Some individuals may employ immutability strategies on purpose, including validation via consensus methods and cryptographic security measures.

Although the majority of blockchains are regarded to be impenetrable, there are flaws. Private keys are taken when a network is compromised in a cryptocurrency theft. This flaw affects permissioned blockchains as well because the networks connecting users to the service rely on security safeguards that can be disregarded.

User data theft and account hacking are also possible, similar to enterprise-level data breaches.

The possibilities of this technology might seem limited and only related to storing information securely via the discussed properties, but the true potential of blockchain is unlocked when a fundamental element of Web 3.0, the *Smart Contract*, comes into play.

1.2. Smart Contract

1.2.1. What are the smart contracts

Smart contracts are programmes based on the if-then approach, that is, they are executed when certain conditions occur. They are the resources that are relied upon to exclude third parties from every operation performed within the blockchain in which they are running. Thanks to these contracts, it is possible to automate each operation not only maintaining security, but moreover increasing efficiency, reducing risks and lowering costs, all while making every operation transparent to users. [3]

1.2.2. How smart contracts work

There are various types of programming languages to write new smart contracts, the most used is *Solidity*, developed in order to operate on the Ethereum blockchain. Every developer is able to create a new smart contract for personal benefits, however, every contract involves more independent parts that might not know each other, consequently, do not trust the other. Every smart contract defines who can interact with it, when and what input result in which outputs. The result of that is the transition from a state in which an event could happen as desired, a sort of *probabilistic state,* to a state in which we are certain of future events depending on certain occurring conditions, named *deterministic state.*

1.2.3. Smart contract benefits

Most of the times that someone effects a purchase online, or that a digital interaction occurs, the parts involved – which can be two or more – does not know each other. In this way comes to life what is called *counterparty risk,* in other words you have to trust someone else you do not know to make sure that everything works fine. In centralized finance the risk is removed/reduced thanks to the intervention of centralized institutions like banks, which acting as a middleware in every transaction, make sure that the trade is successful. From the solution to a problem, other ones are born, and these are resolved by smart contracts thanks to their properties. [4] The contracts advantages are indeed:

- **Security**: Since smart contracts are executed on a decentralized blockchain, they do not constitute a central entity potentially attackable like banks, or they cannot be bribed since all information contained onto the blockchain cannot be modified and are easily verifiable from every user.
- **Reliability**: A bank could fail at any given moment, on the contrary the blockchain technology cannot, other than that every contract is examined and validated from every node on the chain, becoming tamper-proof.
- **Equity**: since there is no central entity, no one can exploit its position for personal gain such as transaction fees.
- **Efficiency**: the absence of an intermediary and the fully automated process mean that the time to complete the transaction is kept to a minimum.
- **Costs:** smart contracts create independence by removing the middlemen from every operation, regardless it is a simple transaction or a more complex operation. Since the smart contract is automated and there is no more a middleman, it is removed any fee usually given to the third party. Other than making the smart contracts cost effective, this streamlines the process.

1.2.4. Smart contracts use cases

When discussing smart contracts, it's common to use the example of an insurance policy that is linked to the blockchain. In fact, the resolution of a claim relates to a particular occurrence; whether or not the smart contract is carried out depends on the circumstances and the input data. By doing that, the friction brought on by intermediaries' involvement is eliminated. Let’s take the example of crop insurance. A farmer has an insurance policy that prevents him to lose all his money in case of a storm or particularly bad weather conditions. If one of these events happens, then the smart contract is automatically triggered and the payout occurs in a transparent way and without friction of any kind, this is allowed by the way smart contracts are written, with conditions that leave no room for interpretations. Other than that, this instrument reduces transaction costs during the processing of claims. [4]

Another example is **Real Estate**: As in almost every aspect of the traditional finance, the traditional real estate system has a third party one must refer to, in order to verify and validate all documents and the bureaucratic part that follows. With the advent of smart contracts every document can be stored and validated digitally into the blockchain. The advantages are not hard to find, since it is saved all the time needed to sign documents and verify them. On top of that, the blockchain is completely transparent and allows everyone to examine the contracts.

Use cases that are only starting to arise:

- **Digital right management**: To organize an event, the digital right industry requires a number of stakeholders. One of the problems is the relationship between copyright and payment %. By employing ownership restrictions in the blockchain system, smart contract-based technologies ensure that royalties flow to the correct recipients.

- **Mortgages**: With the help of this newly developed technology, mortgage transactions can be validated without the aid of attorneys or other third parties.
- **Election voting**: By establishing a secure environment, smart contracts could lower the possibility of voter fraud. A smart contract provides ledger protection for each vote. These are exceedingly challenging to decode because of the encryption. Additionally, smart contracts might boost voter turnout. There is no need to go to a polling place with a smart contract-powered online system.

1.2.5. Smart contracts limitations

One of the positive features of blockchain is that it is isolated from any other network, so as to prevent external attacks. The problem of having no connection to the outside world arises when smart contracts come into play. So far, we have talked about how these contracts are executed depending on whether certain events occur in the outside world. But just as a computer cannot have information without an internet connection, smart contracts become extremely limited without knowledge and connections from the outside world. Taking the example of insurance, if one considers a user with a policy that defends him against particular weather events, a smart contract can never know, nor can it verify, whether such conditions have occurred or not. This led to the development of a technology used to communicate with the outside world: the oracles.

The blockchain is designed to be deterministic, this is the main reason for which smart contracts cannot interact in any way with the external world, whether that is connecting to internet or to an API, indeed if we repeat the same query to one of the two, we cannot be sure that the responses would be the same.

1.3. Oracles

1.3.1. What are oracles

Oracles are the element of the system that allows the communication between blockchain -which is by definition a closed system- and the external world. Acting as a middleware between the two, it gives the chain the possibility to be aware of every event/element external to the blockchain environment, needed from the smart contracts to fulfil their functions by communicating with any off-chain system, from web APIs to cloud providers, other blockchains, etc. A complete list of the oracle functions it is reported here, from the Chainlink website [5]

Oracles perform a number of crucial tasks:

- **Listen**: keep an eye out for any incoming user or smart contract requests for off-chain data by keeping an eye on the blockchain network.
- **Extract**: Obtain information from a variety of external systems, such as off-chain APIs located on external web servers.
- **Format**: convert information acquired from external APIs into a blockchain readable format (input) or transform blockchain information to be compatible with an external API (output).
- **Validate**: provide a cryptographic proof utilizing any combination of data signing, blockchain transaction signing, TLS signatures, Trusted Execution Environment (TEE) attestations, or zero knowledge proofs attesting to the execution of an oracle service.
- **Compute**: For the smart contract, carry out some kind of secure off-chain computation, like determining the median from numerous oracle submissions or producing a verifiable random number for an online multiplayer application.
- **Broadcast**: To provide data and any associated proof on-chain for consumption by the smart contract, sign and broadcast a transaction on the blockchain.

- **Output** (optional): After a smart contract has been executed, send data to an external system, such as transmitting payment instructions to a conventional payment network or launching activities from a cyber-physical system.

For the oracle system to accomplish the aforementioned tasks, it must broadcast data, send proofs, extract blockchain data, and potentially do computations on the blockchain while simultaneously operating both on and off the blockchain connection (to listen for requests). The off-chain component handles request processing, retrieves and formats external data, sends blockchain data to external systems, and performs off-chain computing for improved smart contract scalability, privacy, security, and other features.

1.3.2. Oracles design patterns

1.3.2.1. Immediate-read oracle

Immediate-read oracles provide data that is only needed for a rapid decision, such as "what is this student's grade average?" if someone wants to query this kind of data, he typically does "just-in-time," which implies that the lookup is performed only when the information is required.

Dial codes, academic certificates, institutional memberships, airport identification, and other oracles are examples.

1.3.2.2. Publish-subscribe oracle

An oracle that effectively provides a ondemand service for data that is expected to fluctuate (perhaps on a regular and frequent basis) is either polled by an on-chain smart contract or watched for alterations by an off-chain node. The publish-subscribe model is used for a variety of purposes, including weather data, price feeds, economic or social statistics, and traffic data.

1.3.2.3. Request-response oracle

The most difficult category is request-response, in which the data space is too huge to be contained in a smart contract and users are only expected to access a

small percentage of the whole information at a time. It is also a potential commercial approach for data providers.

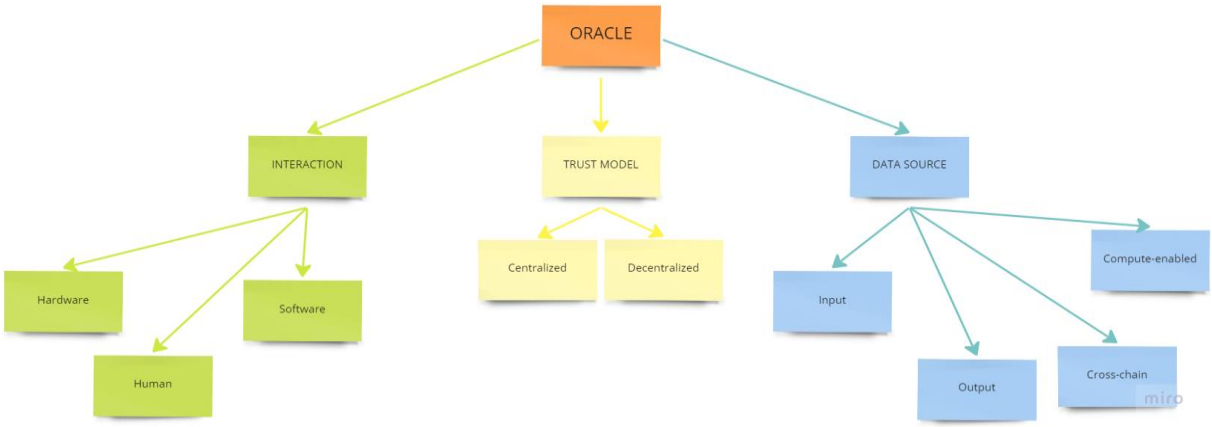
1.3.3. Oracle types

Although connecting a specific blockchain to the outside world remains the primary function of a blockchain oracle, there are other subcategories of oracles that can be distinguished based on their input, output, and other factors.

- **Input oracles**: above all other types of oracles, at least in terms of recognition, there is the input oracle, which takes off-chain data and transmits all the information retrieved into the blockchain network, so that a smart contract can then elaborate it. An example and a common use case for this type of oracle is that they provide price feed and financial market data to blockchains, useful for everything that concerns Decentralized Finance.
- **Output oracles**: these are the oracles that perform the inverse operation with respect to the input oracles, they allow a smart contract to send commands to off-chain systems, so that when they receive one, they will execute a predefined action. One can think of the unlocking of software installed on a computer. Once the payment has been made and the nodes in the chain are able to verify it, the oracle output is able to make the smart contract communicate with the software lock to make it unlock and make the software usable by that machine.
- **Cross-chain oracles**: cross-chain oracles are another kind of oracle that can read and write data between various blockchains. Data and assets can be moved across chains using cross-chain oracles, allowing them to exist outside of the native ledger they were issued on.
- **Compute-enabled oracles**: Compute-enabled oracles are a new category of oracle that smart contract applications are increasingly using to deliver decentralized services that are hard to perform on-chain owing to technical, legal, or economical limitations. They leverage safe off-chain computation to do this. This may involve generating zero-knowledge proofs to create data

privacy, utilizing keepers to automate the executing of smart contracts when predetermined events occur, or using a Verifiable Randomness Function to give smart contracts a tamper-proof and provably fair source of randomness.

It is also feasible to differentiate between software oracles and hardware oracles based on where the data we are collecting came from. The first one uses data from any online source, such as websites, servers, or online databases. This is where the two vary clearly. While the latter relies on physical technologies, such as barcode scanners, electrical sensors, alarms, etc., to transfer information to the chain by turning actual data into digital values.



1.3.4. The oracle problem

According to Caldarelli and Ellul (2021), almost two thirds of DeFi hacks were possible because of oracle exploitation.[5]

Smart contracts are deterministic, so every information that oracles deliver determine their outputs. This brings to the fact that oracles information must always be correct in order to make the whole system work. Here it is raised the main problem of the blockchain's architecture, the centrality of the oracle. Being the only point of contact between blockchain and off-chain elements causes what in information system security is called a single point of failure – but, as reported

[5] Caldarelli, G., & Ellul, J. (2021). The blockchain oracle problem in decentralized Finance – a multivocal approach. http://dx.doi.org/10.3390/app11167572 , last access 26/09/2022.

in an article published on the Chainlink site "the entire point of a smart contract is to achieve determinism through technological enforcement of the contract's terms as opposed to probabilistic execution carried out by human enforcement. To achieve this end, the blockchain cannot have any single point of failure."[6] Since we have only one centralized oracle, which is the only element of the system giving information to the blockchain, if it goes offline, the smart contract will not be able to retrieve the information it needs for execution, which leads to improper execution. Even worse than that, if an information is inserted wrongly, even if it is done without malice, it cannot be removed from the system[7] and could be used for bed intentions or directly lead to unwanted events caused by the wrong output delivered by the smart contract, e.g. in insurance paying a claim which should not be paid – this refers to the problem of a deterministic state known as *garbage in, garbage out* where the insertion of bad inputs in the system corresponds inevitably to bad outputs. Another critical issue are hackers, that knowing this weakness within the system, try to modify information before entering the chain. It takes just one piece of data modified at their advantage and the whole system would be vulnerable to information or asset stealing; that could bring to loss of large amount of money both for the blockchain and the users, other than the loss of the platform's reliability.

Given this information, implementing a centralized oracle inside the blockchain architecture brings to a lot of risks both for the chain and its users, such as being vulnerable to DDoS attacks, bribes, and downtime. Nonetheless, this model is not scalable and goes against the blockchain's will of having a decentralized infrastructure.

Choosing the right oracle is the most important decision for blockchain developers, and their decision should verge to a decentralized oracle.

---

[6] Chainlink, https://blog.chain.link/what-is-the-blockchain-oracle-problem/?_ga=2.256548808.253918674.1659424240-575868723.1659424240 , published on August 27, 2020, last visited on October 7, 2022

[7] The blockchain has the property of being immutable, so every information written on it will remain there whether it is right or wrong.

1.3.5. Decentralized oracles

There is no purpose in having a blockchain – decentralized and reliable system – and then implementing a centralized oracle. If end-to-end determinism of smart contracts is to be preserved, this feature must also apply to the oracle. Oracles need to establish the same security and dependability assurances as a blockchain, but in a different way given their various variances, in order to overcome these drawbacks. The ideal solution must be impervious to all attacks while yet being transparent and unambiguous about how everything operates.

What drawbacks do blockchain decentralized oracles have?

- **Third-party collusion**: Because decentralized oracles depend on a variety of outside sources to get information, they are vulnerable to third parties working together to falsify data or commit fraud.
- **Long duration**: Compared to centralized blockchain oracles, it takes more time to gather information from various sources and reach a consensus on the result. The speed of each network oracle will be important if all of the oracles must come to an agreement quickly.
- **Cost**: Although decentralized oracles are regarded as excellent for protocols with numerous functions executing at various times, they demand a significant infrastructure and upkeep expenditure.

1.3.6. Hybrid smart contracts

Oracles enhance the capabilities of blockchain networks by giving users access to all external resources needed to implement advanced hybrid smart contract use cases that go beyond simple tokenization. Similar to how the internet fundamentally altered how information is distributed, oracle-powered hybrid smart contracts are fundamentally altering how society distributes value and upholds legal obligations.

This type of contract is commonly referred to as hybrid smart contract since it combines two components: the smart contract, which is implemented on the

blockchain, and the Decentralised Oracle Network (DON), an off-chain service that is executed outside of the blockchain and serves the smart contract.

All services outside the blockchain can now interface with it and vice versa in a secure, trustworthy, scalable, confidential, and customizable way thanks to the DON, enabling new functions that were not previously possible. [6]

1.3.7. The most famous oracles

Numerous initiatives are attempting to address these issues in various degrees of decentralization, by integrating advanced attack-prevention measures, reducing reliance on a single trusted middleman, and by implementing various incentive mechanisms.

- **Chainlink**: the leading decentralized data oracle by far. It has a market cap of 7.6 billion dollars and its total value secured is $20,245,189,593.97[8]. With modularity in mind, Chainlink seeks to create a fully decentralized network of oracle nodes that is compatible with Hyperledger, Ethereum, and Bitcoin. Every component of the Chainlink system is upgradeable. The basic goal is to create a secure environment where clients and nodes can trade oracles. Since performance and reputation are made public, good behaviour is encouraged, while bad behaviour results in consequences. On-chain data aggregation from oracles is what they are currently doing, but they want to transfer it off-chain with a clever architecture.
- **Provable**: previously called Oraclize is a blockchain oracle service for modern DApps. It provides a reliable connection between smart contracts and Web APIs.
- **Band protocol**: it is the closest competitor to Chainlink and the second most valuable decentralized data oracle.
- **UMA**
- **Witnet**

[8] Chainlink, https://chain.link/ (last access: 14/09/2022)

- **DOS network**
- **DIA** (decentralized information asset)

1.4. Oracle use cases

1.4.1. Decentralized finance

Decentralized Finance, also known as DeFi, is the first and most common use case for hybrid smart contracts that are powered by DON at the moment. DeFi offers a decentralized, permissionless, non-custodial, and censorship-resistant alternative to the current dysfunctional traditional banking system and is perhaps, the product market fit of blockchain technology. What is less well known, though, is that the existence of DONs is what made the DeFi ecosystem viable, since they are used in order to access financial data about assets and/or stock market/markets in general.

By enabling individuals, businesses, and merchants to perform financial transactions through new technologies, decentralized finance eliminates middlemen. DeFi makes use of connection, software, hardware, security protocols, and peer-to-peer financial networks.

People can lend, trade, and borrow using software that logs and validates financial transactions in distributed financial databases from anywhere there is an internet connection. A distributed database collects and aggregates data from all users and utilizes a consensus process to verify it, making it available from different locations.

By allowing anyone to utilize financial services wherever they are, regardless of who they are or where they are located, decentralized finance eliminates the necessity for a centralized finance model. Through individual-focused trade services and personal wallets, DeFi applications provide consumers greater control over their finances.

One of the main tenets of DeFi is the use of peer-to-peer (P2P) financial transactions. When two parties agree to exchange cryptocurrencies for goods or services without the involvement of a third party, this is known as a P2P DeFi transaction. In DeFi, peer-to-peer lending can satisfy a person's desire for a loan. An algorithm would connect peers who concurred with the lender's terms, and a loan would then be granted. Through a decentralized application, or dApp, P2P payments are made and proceed in the same way as blockchain transactions.

Using DeFi enables:

- **Accessibility**: A DeFi platform is accessible to anybody with an internet connection, and transactions can take place anywhere in the world.
- **Low transaction costs and high interest rates**: Using DeFi networks, any two parties can directly negotiate interest rates and make loans.
- **Security and Transparency**: Smart contracts recorded on a blockchain are open for everyone to study, and records of transactions that have been performed are also available, but they do not identify your name. Because blockchains are immutable, they cannot be altered.
- **Autonomy**: DeFi platforms are independent of any centralized financial institutions, making them impervious to failure or misfortune. DeFi protocols' decentralized structure significantly reduces this risk.

Stablecoins, for example, are a type of token that has its value strictly bonded to the value of a real-world asset through an algorithm (*non-collateralized stablecoins*), or through collateralization equal to the value of the stablecoin on the market (*collateralized stablecoin*). (Tether (USDT) is bonded to USD and its value tries to always match the equation 1 USDT = 1 USD).

1.4.2. Dynamic NFTs and gaming

Non-Fungible Tokens, or NFTs, are digital tokens on a blockchain that each represent something unique, such as a digital piece of art, a special in-game item, rare trading card collectibles, or any other distinct digital/physical asset.[9]

Each unit of fungible assets is identical, interchangeable, and divisible. The US dollar, Bitcoin, are examples of fungible assets that are utilized every day. As opposed to fungible assets, non-fungible assets are made up of completely distinct units. Real estate is a non-fungible example since each property differs from the other in terms of layout, size, location, zoning, and valuation.

By using blockchain networks like Ethereum to identify special physical and/or digital assets, NFTs build on the idea of non-fungibility. A public blockchain is used to confirm and monitor NFT ownership, enabling users to trace any NFT's origins all the way back to its genesis. So, the easiest way to think about NFTs is as a "certificate of authenticity" given by the original author on the blockchain, which offers cryptographic evidence that the holder of an NFT is the true owner of the legitimate asset it is related to.

So, finance is not the only use case for oracles - even if one can argue that NFTs should have a place in the finance sector-. They can enable e.g., dynamic NFTs (we can make them dynamic through the combination of smart contracts and oracles, like change their appearance depending on the weather or another event in the external world). Other than that, it is possible to use compute oracles and their verifiable randomness function to assign random traits to NFTs or to select a winner in an NFT-drop contest. Last but not least there are gaming use cases, where the VRF is used in order to make the appearance of random loot boxes unpredictable or to randomize matchmaking.

[9] Chainlink, https://chain.link/education/nfts , last updated on September 14, 2021; last access October 12, 2022

1.4.3. Insurance

The traditional insurance system takes a very long time to process claims. In a standard contract there is ambiguity, and every stakeholder can manipulate things to bring everything in his favour. By using smart contracts and input oracles, we can verify the occurrence of insurable events during claims processing, on the other hand with the help of output oracles, insurance smart contracts have a way to provide payouts on claims using other blockchains or traditional payment networks. [7]

An example of this use case is travel insurance, the plan is to use a smart contract created on the Ethereum blockchain to automatically repay passengers whose flights or trains were delayed. From this situation both the insurance company and the customers would benefit, as the former might spend less on resources typically used for processing claims, and the latter would get money as soon as the conditions are met.

A benefit of this approach comes from the fact that anyone is able to review the smart contract. In other words, the consumer signing a policy would clearly comprehend the terms of the contract (even though, at the time, he or she requires learn certain programming abilities to grasp the smart contract code). As a result, it would be simpler for him/her to compare policies. Additionally, since implicit trust would be assured by the smart contract, the decision of a policy would no longer be dependent solely on how much a person trusts a certain organization, but rather on objective facts.

1.4.4. Enterprise

Every business is able to link their servers and entire backend system to any blockchain network owing to cross-chain oracles operating as a safe middleware. A company might indeed read and write from every blockchain by being connected to one oracle that can access possibly dozens of them. This enables them to offer

smart contract services to users who desire them without having to independently integrate with each chain.

Chapter 2

Oracles' security

Blockchain oracles are agents that retrieve data from the outside world because the execution environment of blockchain is cut off from it. Oracles are off-chain elements that might be sources of failure in entire blockchain-based systems, despite the fact that blockchain is considered to be extremely dependable. It is still unknown whether blockchain oracles are reliable. Nobody, not even the finest smart contract auditors, is able to completely forecast what will happen when smart contracts are deployed. Given that smart contracts hold billions of dollars in assets, it is reasonable to think that the most sophisticated hackers are continuously hunting for security flaws to exploit and profit from.

The security of oracles is of paramount importance in the implementation of blockchain-based systems. Oracles are the bridge between the blockchain and the external world, providing external data to smart contracts, enabling them to interact with the real world. They are essential for the execution of various decentralized applications (dApps)[10] and their security is vital for the overall security of the blockchain network.

In this thesis chapter, we will be discussing the importance of a secure oracle in a blockchain environment and the implications of a security breach. We will also be presenting various known attacks against oracles, such as the 51% attack and other, and how they can be mitigated. Through our analysis, we aim to emphasize the importance of considering the security of oracles when implementing blockchain-

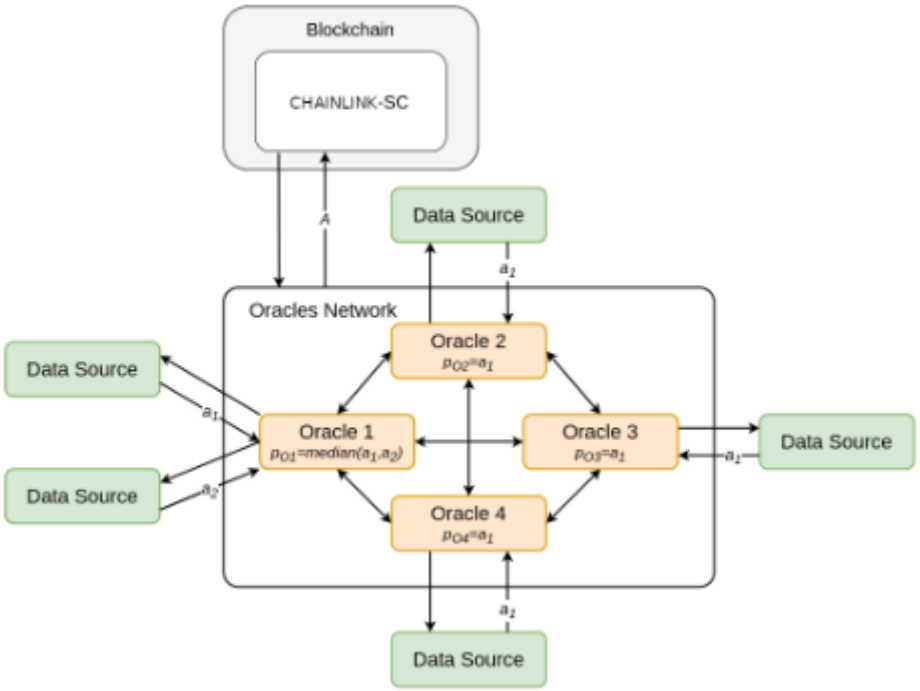
[10] A DEX is a digital marketplace where users can buy and sell cryptocurrency without the need for a central intermediary, like a traditional exchange. Instead, trades occur directly between users through smart contracts on a blockchain network. One popular example of a DEX is Uniswap, which is built on the Ethereum blockchain and allows users to trade a wide variety of ERC-20 tokens. Another example is Binance DEX, which is built on top of Binance Chain and allows users to trade various crypto assets with high speed and low fee.

based systems and contribute to the ongoing research efforts to improve their security.

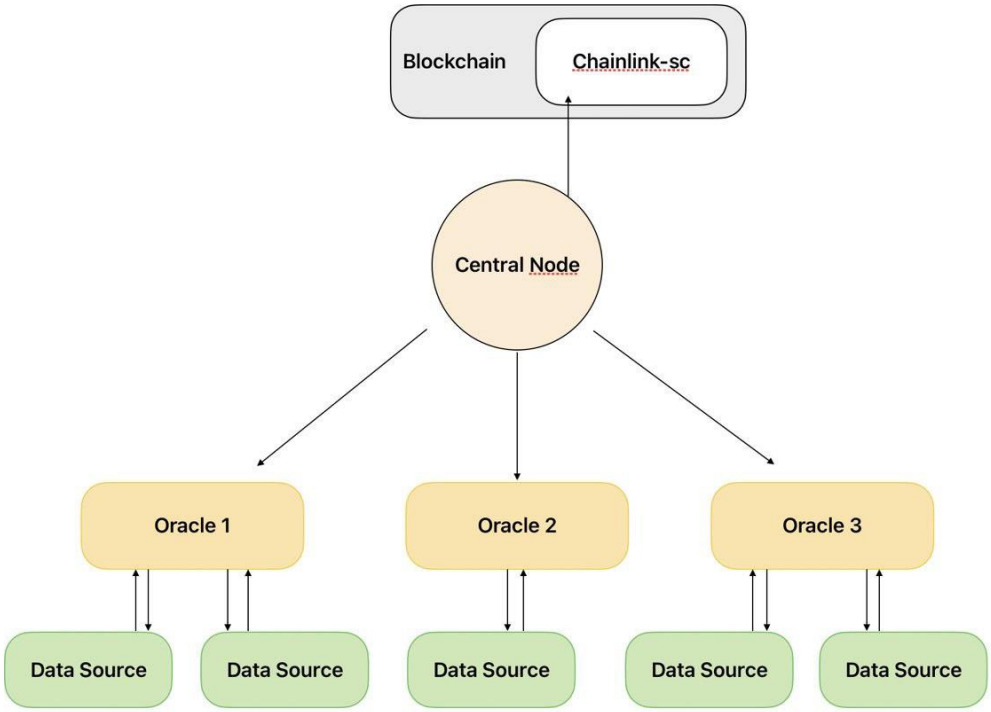
2.1. Oracle consensus

Multiple oracles reach a consensus to produce a final answer that is then posted to the blockchain. The blockchain receives direct external data fetches for a single oracle mechanism. Different platforms utilize different consensus protocols to choose the outcome for numerous distributed oracles.

Multiple oracles in Chainlink utilize a K-out-of-M threshold signature to agree on the answer that will be accepted. To accept a value as the answer, for instance, a 3-out-of-4 signature system requires at least three or more oracles out of four oracles to sign on the same value. Other than that Chainlink uses a reputation system for its oracles, where the oracles that have a history of providing accurate data are given a higher reputation score. The oracles with the highest one, are then used to provide data to the smart contract.



On the other hand, Provable uses a different approach where data is collected from multiple oracles, but then it is processed and verified by a central node. This central node would be the one that provide the data to the smart contract.



In both cases, the consensus mechanism is used to ensure that the data provided to the smart contract is accurate and can be trusted. It's important to have a robust consensus mechanism in place because it helps ensure the integrity of the data provided to the smart contract, which in turn can help ensure the proper functioning of the contract.

2.2. DeFi exploits and consequences

2.2.1. Code in production cultures

Many DeFi projects, led by Andre Cronje, the creator of Yearn Finance[11], adhere to the test-in-production mentality rather than maximizing security and testing to hasten the pace of product development. Audits for every release will greatly increase the time needed to launch any product updates. Developers may iterate much more quickly with DeFi, pushing the envelope for financial innovations, which is one of its primary competitive benefits. But not every project can afford to undergo audits, particularly if it hasn't yet gained much traction. Hackers still succeed in exploiting some projects despite many audits, indicating that audits might not be enough to stop all hacks.

2.2.2. Sloppy coding and insufficient audits

Many project teams feel under pressure to move quickly and use short corners to get their products to market faster in a bull market. Some people may choose to undertake audits only a few months after the items are online in order to gain the first-mover advantage. There are also a lot of "forks," or fresh projects that borrow code from more established ones. They are launched without having a thorough understanding of how the code operates and are viewed as a quick way to get money, which leads to several exploits.

2.2.3. Rug pull (inside jobs)

In the DeFi industry, projects frequently begin with anonymous teams. Due of an unstable regulatory environment, some people use this to avoid regulator inspection. Others, who have negative motives, have opted to remain nameless.

[11] Yearn is a decentralized suite of products helping individuals, DAOs, and other protocols earn yield on their digital assets. https://yearn.finance visited on 19/01/2023.

Numerous times, anonymous teams have carried out an inside job and purposefully left a flaw that is later used to steal from unwary people. Since the first cryptocurrency, Bitcoin, was also created by an anonymous person, the crypto community is not hostile to projects with anonymous creators. Users judge projects based on the code produced, not on the identity or location of the creators. This is consistent with the open software decentralization ideal. Regardless of ideals, the likelihood of no remedy in the event of an exploit on a protocol developed by an anonymous team is significant because it is difficult to determine the real-world identify of the developers. [8]

2.2.4. Oracles attacks

Asset prices are necessary for DeFi protocols to operate properly. For instance, a lending procedure requires knowledge of the asset price to determine whether to liquidate the position of the borrowers. Therefore, oracles may be heavily manipulated because they are a necessary component of the DeFi system.

2.2.5. MetaMask attack

MetaMask is a browser extension that acts as a digital wallet for storing and managing Ethereum-based cryptocurrencies, as well as a portal to the decentralized web (also known as Web3) built on the Ethereum blockchain. It allows users to easily interact with Ethereum-based decentralized applications (dApps) and smart contracts on the web, without the need for a full Ethereum node.

It is not surprising that Metamask has turned into a main assault target given that it serves as the primary interface for all Ethereum applications. Consensus' security measures have been meticulous, and there haven't been any significant exploits to yet. There were a few high-profile attacks, though: $8 million was lost from the founder of Nexus Mutual's personal wallet, and $59 million from the admin MetaMask account of the EasyFi project.

2.3. DeFi security exploits

2.3.1. Reentrancy attack

The ability to use and call the code of other external contracts is one of the capabilities of Ethereum smart contracts. Additionally, contracts frequently handle ether, and as a result, frequently transmit ether to different external user addresses. The contract must make an external call in order to call other contracts or deliver ether to an address. Attackers may take advantage of these external calls to force the contract to run more code, including calls back to itself (using a fallback function). The contract is thus "re-entered" by the code execution. Such attacks were employed in the infamous DAO hack. When a contract sends ether to an unidentified address, this attack may take place. Attackers can craft a contract with harmful code in the fallback function and send it to an external address. Therefore, the malicious code will be executed whenever a contract delivers ether to this address. Typically, the malicious code runs a function on the weak contract and carries out actions that the developer did not want. Re-entrancy (from the fact that the external malicious contract calls back a function on the weak contract and "re-enters" code execution at any point on the weak contract) is the technical term for this type of attack.

The real-world scenario mentioned above occurred in 2016 with the DAO (Decentralized Autonomous Organization) on the Ethereum blockchain. The DAO was a smart contract that functioned as a decentralized venture capital fund, where users could vote on proposals for funding projects.

The attacker exploited a vulnerability in the smart contract code, which allowed them to repeatedly call the "split" function of the contract, which transferred DAO tokens from the smart contract to a separate child contract controlled by the attacker. The attacker was able to repeatedly call the function before the smart contract had a chance to update its internal state, effectively allowing them to drain the DAO of 3.6 million Ether (worth around $50 million at the time).

This attack resulted in a hard fork of the Ethereum blockchain, where the community agreed to a new version of the blockchain that would return the stolen funds to their original owners. This incident highlighted the importance of auditing smart contract code for potential vulnerabilities and the need for better security practices in the development of smart contracts.

2.3.2. Flash loan attacks

As long as the borrower repays the loan during the same transaction, flash loans enable users to leverage almost unlimited cash to complete a financial transaction. It is an effective weapon that makes it possible to steer economic attacks that were previously limited by capital requirements. Utilizing chances with flash loans merely requires having the appropriate technique. Flash loans were used in almost all DeFi hacks. In the part after this, we shall investigate it in further detail.

A decentralized, unsecured loan that may be obtained without the help of a middleman is what a flash loan is, to put it briefly. To ensure that the borrower repays the principal and interest, most lending protocols are constructed on (over)collateral methods. A borrower can obtain a quick loan without offering any collateral, though. Flash loans take place in one block without any collateral. After obtaining out the loan, the borrower fulfills the loan's objectives and pays it back with the accumulated interest. One block is needed to complete the entire operation. If someone takes out a loan against some assets in block 2000, they must pay it back with interest in the same block. It cannot be paid back in block 2001. The transaction would be abandoned if the smart contract that provided the flash loan did not get the repayment in the same block. That is a flash loan, which is a quick loan with a limited duration. Malicious actors can easily take out big flash loans based on the flash loan's mechanism, utilize those money to influence the market, or use different DeFi protocols to make significant gains, usually at the expense of regular investors and platform users.

2.3.2.1. Possible attack process

A flash loan attack is a type of attack that takes advantage of the unique properties of flash loans, which are short-term, high-value loans that must be repaid within a single block. Flash loans are typically used for arbitrage and liquidity provision, but they can also be used for malicious purposes.

The process of a flash loan attack typically involves the following steps:

- The attacker borrows a large amount of assets using a flash loan from a decentralized finance (DeFi) platform.
- The attacker then uses the borrowed assets to execute a trade or other transaction that takes advantage of a vulnerability in the system. For example, the attacker might use the assets to manipulate the price of a token on a decentralized exchange (DEX).
- The attacker then repays the flash loan, often profiting from the transaction they executed using the borrowed assets.
- The attack can be done with a very low risk because the flash loan is temporary, the assets will have to be returned, but the attacker has already taken advantage of the vulnerability in the system.

It's important to note that the flash loan attack is a complex and sophisticated type of attack that requires a deep understanding of the underlying system and its vulnerabilities. Flash loan attack prevention is a current topic of research and development in the decentralized finance space.

One real-world example of a flash loan attack occurred in August 2020 on the Aave decentralized finance (DeFi) platform. An attacker borrowed over $350,000 worth of DAI (a stablecoin pegged to the US dollar) using a flash loan. The attacker then used the borrowed DAI to buy a large amount of USDC (another stablecoin) on the platform. This caused the price of USDC to increase, allowing the attacker to sell the USDC at a higher price and make a profit. After making the profit, the attacker returned the borrowed DAI, effectively completing the flash loan attack.

This attack demonstrated how flash loans can be used to exploit vulnerabilities in DeFi platforms and how important is the security of such platforms. As a result, this attack raised concerns about the potential risks associated with flash loans and the need for better security measures to protect DeFi platforms from similar attacks in the future.

It's important to note that the flash loan attack is a complex and sophisticated type of attack that requires a deep understanding of the underlying system and its vulnerabilities. Flash loan attack prevention is a current topic of research and development in the decentralized finance space.

2.3.2.2. Defensive measures

By using decentralized pricing oracles like Chainklink and Band Protocol to obtain price feeds rather than relying solely on a single DEX platform, DeFi can best counter this flash loan assault (which is vulnerable to attacks).

To honor the transactions, flash loan attackers take advantage of the DeFi platform's latency. Are there any tools that can help you stop this loop exploitation? Yes, there are certain solutions like OpenZeppelin. DeFi platforms can make use of these tools to find vulnerabilities and exploits in smart contracts as well as track any anomalous behaviour in order to take preventative measures against assaults.

Chainlink move: ChainLink is less vulnerable to "Flash Loan" assaults since it does not rely on a single centralized pricing feed. Flash Loan attacks have cost DeFi projects millions of dollars by allowing astute users to fool protocols into releasing more liquidity than they have staked tokens to sustain. In addition to Flash Loans, manipulating Liquidity Pools to generate arbitrage possibilities has been demonstrated in specific instances, such as the $34 million Harvest Finance attack. Although the attacks mentioned above did not involve Chainlink, they nevertheless highlight the vulnerabilities of an increasingly complicated DeFi environment in which prices are established by various pathways.

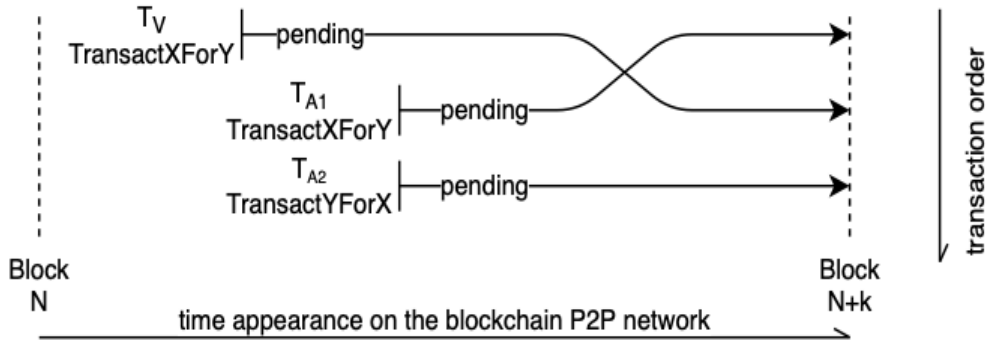
2.3.3. Sandwich attacks

A sandwich attack is a type of front-running that mainly targets the protocols and services used in decentralized finance.

A sandwich attack is a type of attack that takes advantage of the mechanism of atomic swaps, which are a type of smart contract that allows for the exchange of one cryptocurrency for another without the need for a trusted intermediary. The attack is so called because it is said to "sandwich" the intended trade between two malicious trades, thereby allowing the attacker to profit at the expense of the intended trade's counterparty.

The process of a sandwich attack typically involves the following steps:

- The attacker creates a fake order on a decentralized exchange (DEX) with an attractive price for a certain cryptocurrency
- The attacker then creates another fake order on another decentralized exchange with the same cryptocurrency at a higher price.
- The attacker then initiates an atomic swap with the victim, who thinks that they are getting a good deal because of the first fake order.
- The attacker then completes the atomic swap with the second fake order, resulting in a profit for the attacker and a loss for the victim.



For example, let's say the current market price for a certain cryptocurrency is $100, the attacker creates a fake order on DEX A for $90 and another fake order on DEX B for $110. A victim sees the order on DEX A and initiates an atomic swap with

the attacker to buy the cryptocurrency, thinking they are getting a good deal. The attacker then completes the atomic swap with the order on DEX B, resulting in a profit of $20 for the attacker and a loss of $10 for the victim.

It's important to note that this type of attack is not a common occurrence and it's hard to pull off due to the nature of decentralized exchanges where the transactions are transparent and can be tracked. The attack prevention is a current topic of research and development in the decentralized finance space.

2.3.4. Front-end attacks

The technologies used in web attacks include front-end attacks. XSS, CSRF, network hijacking, and phishing assaults are some of the typical front-end attacks. Here is further information about these attacks:

- **XSS Attack:** Cross-Site Scripting, often known as XXS, is an assault on websites that have security flaws. In order to launch the attack, the user's browser must be running JavaScript or illicit HTML tags. Traps are set by attackers using scripts. If consumers are not careful, they will be passively attacked while using their browsers. False input forms used to steal cookies, transmit malicious requests, and obtain personal information from users are examples of common XSS attacks. Escaping HTML elements and JavaScript, as well as preventing JavaScript from reading cookies, are common techniques for preventing XSS attacks.
- **CSFR Attack:** refers to a hacker who sets up traps to coerce users who have successfully completed authentication into providing unexpected personal information or setting information and other status updates.

2.3.4.1. Real-world front-end attack

A significant frontend attack on Badger DAO resulted in the theft of user assets without the users' permission. More than 120 million dollars in value had been lost in this incident, according to the developer's initial inventory of destroyed assets; the amount taken ranks fourth in recent DeFi attacks! With the use of

bitcoin and BTC-related assets, you can earn yield and prizes on the Ethereum DeFi network known as BadgerDAO. As opposed to keeping your BTC in its current state, the protocol offers options to make larger ROI. In essence, BadgerDAO turns your Bitcoin into a useful resource. From their official website, we can see that they appear to take many precautions to stop money from being stolen, including audits, the Council of White Hats, bug bounty programs, and insurance. However, Badger did suffer a loss. What conclusions can we draw from that?

2.3.5. Phishing attack

Phishing is a form of social engineering when an attacker sends false (for instance, false, forged, or other false) messages to lure human targets into giving sensitive information to the attacker or allowing the attacker to install malware on the victim's infrastructure. Users should take all advised security precautions in order to develop some resistance to it. Admittedly, the antiquated " phishing " assault approach is still prevalent and has migrated to the sphere of NFT and DeFi assets in the developing decentralized network blockchain. It makes use of the user's lack of understanding with the blockchain technology.

Common techniques

- **Legitimate links:** By including trustworthy links in trick phishing emails, many attackers attempt to get past email filters. They can accomplish this by presenting the legitimate contact details of the company they might defraud.
- **Combining dangerous and benign code:** To trick ExchangeOnline Protection, the individual who creates the login page of a false website frequently combines malicious and innocuous code (EOP). The user's login information might be taken by duplicating the CSS and JavaScript from the tech giant's login page.

2.3.6. Price oracle manipulation

2.3.6.1. Oracle failure (random attack)

Users can create alternative monetary assets using the Ethereum-based derivatives liquidity protocol Synthetix. To do this, Synthetix (at the time) used a unique off-chain price feed method to aggregate a number of hidden price feeds and publish the final price on-chain on a regular basis. Users are then permitted to trade long or short against the asset depending on the determined price.

2.3.6.2. Use flash loans to manipulate the price of AMM

It is simple to carry out an exploit if an attacker has access to the price oracle. This is how the complete procedure appears: Attacker raises the price of the borrowed token (or lowers the price of the collateral); price liquidates the security. Attackers frequently use flash loans to influence the price of AMMs to change the spot price of a token before the lender smart contract looks up the token. This is frequently the case with on-chain centralized oracles and decentralized exchanges.

2.3.6.3. Centralized exchange leaks

If the exchange account's private key is stolen, a hacker can simply create collateral tokens and use those to withdraw the entire balance. Until the oracle reflects the new price, they can also borrow fresh, diluted collateral tokens.

2.3.6.4. Arbitrage

For instance, every 24 hours or whenever the price changes by 2%, Chainlink updates the Dai contract. Dai's price can be adjusted anywhere between $0.97 and $1.03. When the price of Dai is adjusted to fluctuate within a 2% range, losses are still probable.

2.3.7. Front running attack

In general, front-running is the act of redeeming a transaction before a known future transaction takes place by advancing in the execution queue. These advanced robots have the ability to examine smart contract instructions and

functionalities that they have never utilized before in a smart contract in order to extract possible gains and cut off funding. When a malicious user watches a swap transaction after it has been broadcast but before it is finished and rearranges transactions to their advantage, this is known as a front-running attack. A miner or bot will frequently put their own transaction right in front of the one that is currently pending. Front-Running is the practice of watching for the packaging of a typical transaction. By setting a larger gas charge in advance of starting the attack transaction, the front-running robot successfully captures the user's interests. Mempool is a collection of Ethereum transactions that are awaiting block packaging after being broadcast to the network. It is assumed that Front-Running can be put into practice. The leading robot continuously scans the transactions in the Mempool to assess and identify targets that can be attacked.

The natural circumstance where the oracle is the first to be aware of crucial data uploaded to the blockchain to run a smart contract causes front-running attacks. It may be argued that the early detection of data gives the Oracle data provider a significant edge.Fraudulent activities may be made possible if it is known that an asset price will be delivered to a specified platform at a specific time.

Furthermore, front-running assaults become more and more possible due to block-time, congestions, the use of bots, and automated processes. The front-running attack is well-known in legacy banking, but it can only be used by a small group of persons (insiders) who have access to information that is not in the public domain. Anyone who recognizes the possibility of information being used for a front-running attack may be a prospective attacker in decentralized finance because information is transparent. A well-known illustration of a front-running assault is the Terra Money exploit. A stable coin called Terra has the potential to be used on several different blockchains, including Ethereum and Solana. A front-running assault on Terra's oracle, which was created to be secure and trustworthy, occurred in 2019. Three processes are used to register the price of external assets in the Terra oracle price feed. The pre-vote phase (N-2) is the first and is when the price of an external asset is put forward as a feed. The second step is the voting

phase (N-1), where the plan might be rejected if there are not enough votes. The proposal is finally registered on the blockchain during the confirmation phase (N), which is the third. There are 24 blocks between the pre-voting phase and the confirmation phase because each stage in this system lasts for 12 blocks. This indicates that the price that is accepted at time N is that of time N-2. The hack was made easier by the fact that Terra coin exchanges were free at the time of the attack in an effort to encourage platform trading. Then, in August 2019 (Figure), a perpetrator discovered a little (2%) difference between the spot price and the oracle price. The zero costs concept was then used to enable asset trading at a reduced price. Terra Money was compelled to start charging two fees after that: fixed and proportional. Front-running attacks become problematic as a result of a little price difference of 1–2% being fully absorbed by the fees.

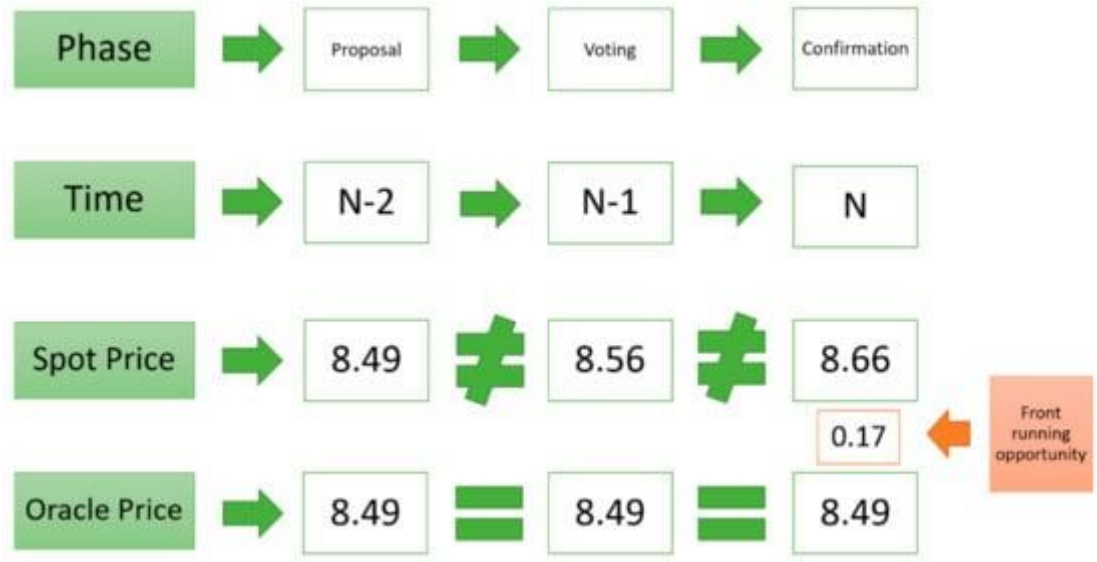


Figura 2.3.7.1 example of a front-running opportunity

The front-running assault can also be carried out via taking advantage of the blockchain's fee system, which gives higher-cost transactions priority. It is also feasible to look at pending transactions that have not yet been confirmed because blockchain is publicly auditable. Prices will undoubtedly be impacted, for instance, if we see an exchange of thousands of DAI for another asset in the transaction pool. We can gain from that prioritized operation if we then carry out a transaction and pay a greater fee so that our transaction is mined ahead of the swap. Although it

incorporates oracles, this "Miners Extractable Value (MEV) Problem" cannot be regarded as an oracle manipulation.

2.3.8. Impermanent loss

Transient losses happen when you lend money to a liquidity pool, and the price at which you deposit the item differs from when you deposit it. Your losses will be more variable the more this variance there is. The dollar value at the time of withdrawal in this instance is less than the dollar value at the time of deposit, which is the definition of a loss. Because the loss is only realized after you remove tokens from the liquidity pool, it is known as an impermanent loss. However, the losses are irreparable, even though the fees you make could offset them. When you mine for liquidity, losses of this nature happen. This loss results from the token's price disparity; when prices stabilize, the loss goes away. Impermanent loss, in essence, happens when depositing assets into an automated market maker (AMM) and then withdrawing them causes a loss as opposed to if you had just kept your assets in your wallet. Because losses are not recorded until funds have been taken from the liquidity pool, impermanent losses are just that—temporary. Any possible losses up until that point could be minimized or eliminated entirely depending on the direction of the market. You must first comprehend how the liquidity pools employed by AMM-based decentralized exchanges like Uniswap, SushiSwap, or PancakeSwap operate in order to comprehend how temporary losses arise.

2.4. 51% Attack

Although it hasn't been proven to happen in the most well-known systems thus far, the 51% Attack is undoubtedly one of the most well-known and potent attacks in the context of cryptocurrencies. If a malevolent person controls 51% of the system's distributed computer capacity, then such an assault is possible. With this level of control, the attacker can disrupt the normal functioning of the network and potentially steal funds or damage the network's reputation. The processing power of the Bitcoin network is measured in terms of what is more commonly referred to as hashrate. In other words, it measures the speed at which a computer completes

an operation, such as the time it takes to add new transactions to the blockchain and the likelihood that it will create legitimate blocks. The above-mentioned malicious user would be able to repeatedly send bitcoins to his wallet by reversing the blockchain register, making it appear as though the initial transactions never happened. A blockchain's reliability is determined by the network's consensus, which determines which blocks to accept. An attacker would be able to build longer blockchains faster than trustworthy nodes if, at this phase, he could generate blocks more quickly by abusing more computer power.

They would benefit from this because lengthier blockchains are frequently regarded as more dependable in the context of Bitcoin. Therefore, if a 51% attack took place, there would undoubtedly be a loss of faith in the system and a swift devaluation of the currency, but as was already noted, no such attacks have been validated as of now, at least not for Bitcoin. As we've seen so far, Bitcoin has a decentralized architecture and employs cryptographic techniques to safeguard the system; nevertheless, this is also made feasible by the fact that it is based on the distributed consensus principle.

The process of a 51% attack typically involves the following steps:

- The attacker acquires a significant amount of mining power, either by purchasing or renting it, or by building their own mining rigs.
- The attacker then uses this mining power to control more than 50% of the network's hashrate.
- The attacker can then use their control over the network to prevent new transactions from being confirmed, reverse transactions, and double-spend coins.

A real-world scenario of a 51% attack occurred in January 2019 on the Ethereum Classic (ETC) blockchain, where an attacker was able to double spend over $1 million worth of ETC by controlling more than 50% of the network's hashrate. The attacker used this control to reorganize the blockchain, allowing them to double-

spend their own coins, and in turn, causing a loss of funds for multiple exchanges and individuals.

It's important to note that 51% attack is a rare occurrence, it's expensive and difficult to achieve, and the blockchain networks with more security measures like decentralized mining pools, hardware wallets and multi-sig transactions are less susceptible to these types of attacks.

2.5. Conclusions

In conclusion, the security of oracles is a crucial aspect to consider in the implementation of smart contracts. This chapter has discussed various types of attacks that can be launched against oracles, including flash loan, arbitrage and 51% attacks. It is clear that the use of secure and reliable oracles is essential for the proper functioning of smart contracts and the protection of users' assets. Further research is needed to develop effective countermeasures to protect against these types of attacks and to ensure the security and integrity of oracle-based systems. Overall, the security of oracles is a complex and constantly evolving field that requires ongoing attention and innovation.

Chapter 3

Chainlink and Provable

Decentralized services are those that provide data outside of the blockchain to smart contracts on the blockchain, such as market prices or meteorological information.

Decentralized servers are frequently used to expand the capabilities of smart contracts beyond the data already present on the blockchain.

Due to their ability to access external data from the cryptographic world, decentralized oracles are essential components of smart contracts on the blockchain. Oracle Provable and Chainlink are the two main decentralized Oracle protocol.

Both Chainlink and Oracle Provable are decentralized database protocols, although there are several key differences between them.

Oracle Provable employs a method based on the test to provide reliable data to intelligent contracts. This data verification is done on a central server which could lead to possible attacks to the system.

In contrast, Chainlink gathers and verifies external data before providing it to smart contracts via geographically distributed oracle nodes. It provides a mechanism for rewarding data-accurate oracle nodes, which improves the quality of the data supplied to smart contracts. However, a potential attack on a decentralized oracle network like Chainlink might involve fabricating a large enough number of false nodes to fool the system and provide false information to intelligent contracts.

3.1 Provable

Provable is a renowned Oracle service provider that works on systems such as Ethereum.

Provable does not use a completely decentralized oracle system, such as Chainlink, since they believe it is intrinsically inefficient because all parties involved will be charged a fee and it will take time to get a sufficient number of replies for each request. In addition, a fully decentralized Oracle system necessitates an established data format standard. Instead, Provable's answer is to establish that the data

obtained from the original data source is real and unaltered. To prove this, the data is returned along with an authenticity proof document, which is generated using various technologies such as auditable virtual machines and Trusted Execution Environments.

As a result, the application developer does not need to trust the Provable oracle because it simply needs to check the authenticity evidence. Given a trustworthy data source S, the Authenticity Proof verifies that a given data from S has not been altered with by the Provable oracle that acts as a middleman between the blockchain and S. Furthermore, the data supplier does not need to change the interface of their services to make them blockchain-compatible.

3.1.1 Provable data validation

Provable is designed to act as an untrusted intermediary. Optionally, a request to Provable can specify an authenticity proof. Not all proofs are compatible with all data source types. Analysing the Provable's documentation [9] we found three different methods for data authentication:

- **TLS notary**: The TLSNotary Proof makes use of a TLS 1.0 and 1.1 protocol[12] feature that permits the server, an auditee, and an auditor to share the TLS master key. In this approach, an open-source, specially created Amazon Machine Image serves as the auditor, with Provable serving as the auditee.

- **Android Proof**: In traditional oracle systems, the smart contract relies on a centralized oracle to access off-chain data and send it back to the contract.

[12] This two protocols do not support authenticated encryption but only utilize authentication then encryption with DES, which is now deprecated and considered insecure, or at best 3DES, that will be prohibited after 2023 https://www.encryptionconsulting.com/why-3des-or-triple-des-is-officially-being-retired/ visited on 21/12/2022 .

This creates a potential point of failure and security vulnerability, as the oracle could potentially tamper with the data or be compromised by an attacker. Android Proof addresses this issue by allowing the smart contract to communicate directly with the Android device, which acts as the oracle. The smart contract sends a query to the device, which then retrieves the requested data and sends it back to the contract. This eliminates the need for a centralized oracle and ensures that the data received by the contract is accurate and untampered. A problem found with this feature is that in the it utilizes SafetyNet Attestation API, which is considered deprecated[13].

- **Wolfram Alpha**: The Wolfram Alpha data source type enables direct access to the Wolfram Alpha Knowledge Engine API. This data source expects as sole parameter the string which should be passed to Wolfram Alpha. It will return the result as a string. This data source doesn't support authenticity proofs as returning the whole API response is against Wolfram Alpha Terms of Service. For this reason, Provable recommends using this data source type only for testing.

The next step is going to analyse how Provable validates the accuracy of data, and it does that utilizing common techniques like:

- **Digital signature verification**: Using a private key, the external data source digitally signs the data. The public key is then used to confirm the signature and make sure the data hasn't been tampered with throughout transfer on Provable's decentralized oracle platform.
- **Verification of data origin**: To confirm that the data originates from the intended external data source, Provable's decentralized oracle platform can use methods like IP address verification.

[13] For more information refer to https://developer.android.com/training/safetynet

- **Data integrity verification**: To ensure that the data hasn't been changed during transport, Provable's decentralized oracle platform may employ methods like checksums.
- **Data history verification**: To confirm that the data was generated at a particular time, Provable's decentralized oracle platform may employ methods like timestamping.

The verification of the IP address as a means of assuring the origin of the data, as explained in the second point, can have certain limitations, and is not thought of as an entirely reliable method of doing so, actually, not at all. First off, using techniques like IP spoofing or VPN tunnelling, it is possible to easily fake or manipulate a system's IP address. Additionally, a system's IP address may change over time, making it challenging to rely on it as a means of identifying a specific system.

The recovered data are verified to ensure their veracity in the third step of the Provable decentralized cloud computing platform's operation. A criticism of Provable's decentralized oracle platform is the need for data verification because smart contracts running on the blockchain must be able to rely on the information provided by the oracle. The Provable decentralized oracle platform employs the technique of data integrity verification to make sure that the data were not altered during transfer. Checksums are one method used to ensure the integrity of data. A control code called a checksum is calculated for a set of data. If the data are altered in any way, the calculated checksum for the altered data will differ from the original checksum. As a result, the checksum can be used to confirm that the data was not altered during transfer. The external data source calculates the checksum for the data and provides it along with the original data to the Provable decentralized oracle platform in order to use it to verify the integrity of the data. Therefore, the Provable decentralized oracle platform calculates the checksum for the received data again and compares the result to the checksum provided by the external source of data. If the two checksums match, it means that the data were not changed during transfer and may therefore be regarded as expectable.

3.1.2 Provable oracle price feed implementation

Here's a possible implementation of a Provable price feed oracle:

```
pragma solidity >= 0.5.0 < 0.6.0; //Declaring the Solidity version

import "github.com/provable-things/ethereum-api/provableAPI.sol";

//Importing latest version of provable API

contract BitcoinPrice is usingProvable {

//Contract named BitcoinPrice, UsingProvable refers to the API

    uint public bitcoinPriceUSD;

//bitcoinPriceUSD is the variable created to store the price,
Provable query event that makes a constructor

    event LogNewBitcoinPrice(string price);
    event LogNewProvableQuery(string description);

    constructor() public{
        update();
    }

// callback function to call the smart contract after the output is received
and transfers the result from callback function to the variable assigned

    function __callback(bytes32 _myid, string memory _result) public {
        require(msg.sender == provable_cbAddress());
        emit LogNewBitcoinPrice(_result);
        BitcoinPriceUSD = parseInt(_result, 2); // Let's save it as cents
    }

//passing output string and API string to fetch bitcoin price to our
constructor


    function update() public payable{
        emit LogNewProvableQuery("Provable query was sent,
      standing by for the answer...");
        provable_query("URL", xml("https://min-
api.cryptocompare.com/data/generateAvg?fsym=BTC&tsym=USD&e=Kraken");
    }
}
```

Provable_priceFeed code source 3.1.1

Last query is not encrypted, what about confidentiality? When dealing with an application placed on a public blockchain, we sometimes don't want our data to be publicly available, even if doing so reduces the blockchain technology's audibility. When we wish to obtain data from an authenticated API, this is required. To address this issue, Provable provides the option of encrypting query parameters with the Provable's public key. Only Provable will be able to decrypt the request using its paired private key and use it in a trusted execution environment this way.

3.2 Chainlink

Chainlink is built on the Ethereum blockchain, and it uses a network of decentralized oracles that can access a wide range of external data sources, including off-chain data, APIs, and other data feeds. The Chainlink network is composed of two main components: the Chainlink nodes and the Chainlink contracts. The Chainlink nodes are responsible for collecting and processing external data and sending it to the Chainlink contracts, which in turn, use this data to execute smart contracts.

One of the key advantages of Chainlink is its high level of security. Chainlink uses a decentralized network of oracles, which eliminates the risk of a single point of failure or a central point of control. Additionally, Chainlink uses a reputation system and a reputation token (LINK) to incentivize node operators to provide accurate and reliable data. This helps to ensure that the data being used by smart contracts is accurate and trustworthy.

Another advantage of Chainlink is its flexibility. The network can access a wide range of external data sources, including APIs, web pages, e-mail or even IoT sensors. This makes it possible to use Chainlink for a wide range of use cases, including financial applications, prediction markets, and more.

Despite its advantages, Chainlink also has some drawbacks. One of the main drawbacks is its scalability. The Chainlink network is currently built on the Ethereum blockchain, which has limited scalability. This can result in high

transaction fees and slow confirmation times, which can limit the adoption of Chainlink in certain use cases.

3.2.1 Chainlink data aggregation

Chainlink Price Feeds were created with the specific intention of giving DeFi applications the highest level of price oracle security, dependability, and data quality. Decentralization at the level of the oracle node and data source, the choice of secure node operators and premium data sources, verifiable on-chain performance and dependability, and crypto-economic incentives for security are only a few of the design decisions that result in these qualities.

When talking about Price Feeds in Chainlink we face a different approach then the one of Provable, indeed, there is a three step process that data will pass named three level data aggregation which are:

- **Data source aggregation**
- **Chainlink node operators**
- **Oracle network aggregation**

3.2.1.1 Data source aggregation

The actual data sources that the Chainlink oracles are using to collect price data make up the first part of a Chainlink Price Feed.

Typically, off-chain centralized exchanges like Coinbase and Binance and on-chain decentralized exchange protocols provide raw pricing data (e.g. Uniswap[14]).

[14] Uniswap is a decentralized exchange (DEX) built on the Ethereum blockchain, which allows users to trade ERC-20 tokens without the need for a centralized intermediary. Uniswap uses an automated market maker (AMM) model, which allows it to provide liquidity for trading pairs through the use of smart contracts.

In order to create refined datasets, data aggregators (like CoinGecko[15]) gather raw exchange data from all of these exchanges. These datasets are then refined by removing outliers, screening phony exchange volume, checking for exchange downtime, and many other factors.

In order to avoid data manipulation vulnerabilities and/or volume shift inaccuracies, it is essential to have full market coverage, where a price point represents a refined aggregate of all trading settings rather than a single exchange or even limited group of exchanges.

Chainlink Price Feeds only use data from top data aggregators to guarantee a high level of tamper-resistance and dependability.

In other words, each data source is a refined, volume-adjusted pricing point gathered from all centralized and decentralized exchanges, making it intrinsically resistant to various attack vectors like flash loans or odd deviations.

3.2.1.2 Node aggregation operation

The on-chain response of each individual oracle node operator makes up the second part of a Chainlink Price Feed. These node operators are made up of qualified DevOps teams with experience running mission-critical blockchain infrastructure, which has already secured billions of dollars' worth of on-chain value. They are in charge of managing the Chainlink core program, which is used to gather and disseminate market data on the blockchain.

To reduce outliers and API downtime, node operators in Chainlink Price Feeds gather price data from various independent data aggregators and take the median (middle) value among them. This means that each individual node's answer also

[15] CoinGecko is a cryptocurrency data and research platform that provides a wide range of information on various cryptocurrencies, including their prices, trading volumes, market capitalization, and more. The platform also tracks the performance of cryptocurrency exchanges and provides a variety of tools for analysing and comparing different cryptocurrencies.

represents an aggregate from several data sources, further preventing any single source from serving as a single point of failure. Each individual data source thus not only reflects an aggregated price point from all trading environments.

Operators of Chainlink nodes extract the median value from several data aggregators.

3.2.1.3 Oracle network aggregation

The entire oracle network is the third element of a Chainlink Price Feed. An oracle network describes how a group of nodes collaborate to provide a single reference data point on-chain, which typically entails aggregating all of the nodes' individual responses. When a predetermined number of nodes have answered, the most popular method of aggregation is to take the median of the reported values. In the end, aggregation can be carried out on-chain or off-chain depending on the cost and throughput of the underlying blockchain network.

Chainlink Price Feeds take a median of the responses from many security-reviewed node operators and aggregate them. An on-chain price update is only triggered by responses that meet a certain criteria (e.g. minimum of 14 out of 21 in the example below). This kind of Oracle network aggregation makes sure that, even in the unusual event that a few nodes or data sources were to go down or attempt to engage in malicious conduct, the Oracle network as a whole, maintains high uptime and resistance to manipulation in its transmission of data on-chain.

DeFi apps get industry-grade security and reliability on the pricing data they utilize to determine how to manage user funds by utilizing multiple layers of aggregation in the data source, node operator, and oracle network layers of Chainlink Price Feeds. Because of this, Chainlink Price Feeds have grown to be the DeFi economy's most popular source of safe on-chain price data, securing billions in on-chain value.

With security and dependability architected into all tiers of the Chainlink Network, dApps receiving Chainlink Price Feeds can be certain that their contracts will always execute as expected, creating a stable foundation from which to scale to ensure additional value for consumers.

3.2.2 Off-chain reports

Off-Chain Reporting (OCR)[16] is a crucial step in increasing Chainlink networks' decentralization and scalability.

All nodes in an Off-Chain Reporting Aggregator communicate using a peer-to-peer network. During the communication process, a lightweight consensus procedure proceeds in which each node reports and signs their data observation. A single aggregate transaction is then broadcast, resulting in significant gas savings.

The report included in the whole transaction is signed by a quorum of oracles and includes all of the oracles' observations. We preserve the trustlessness qualities of Chainlink oracle networks by validating the report on-chain and checking the quorum's signatures on-chain.

3.2.2.1 What is OCR?

The OCR protocol enables nodes to off-chain aggregate their observations into a single report utilizing a secure P2P network. The aggregated report is then submitted to the chain by a single node. Each report is made up of observations from many nodes and must be signed by a quorum of nodes. These signatures are validated on the blockchain.

The following advantages result from submitting only one transaction every round:

[16] For a technical deep dive, see the OCR Protocol Paper.

- Overall network congestion from Chainlink oracle networks has been significantly decreased.
- Individual node operators spend a lot less money on gas.
- Because data feeds may support more nodes, node networks are more scalable.
- Data feeds can be updated more frequently because each round does not need to wait for several transactions to be confirmed before a price is confirmed on-chain.

How does it work?

Protocol execution takes place primarily off-chain, via a peer-to-peer network between Chainlink nodes. The nodes elect a new leader node on a regular basis, which drives the rest of the protocol.

The leader solicits freshly signed observations from followers on a regular basis and compiles them into a report. It then sends this information back to the followers, asking them to check its accuracy. If a quorum of followers signs and returns a signed copy of the report to the leader, the leader compiles a final report with the quorum's signatures and broadcasts it to all followers.

According to a randomized timetable, the nodes attempt to transmit the final report to the aggregator contract. The aggregator confirms that a quorum of nodes signed the report before exposing the median value to users as an answer with a block date and a round ID.

To eliminate any single point of failure during transmission, all nodes monitor the blockchain for the final report. If the chosen node's transmission is not confirmed within a certain time frame, a round-robin procedure is activated, allowing additional nodes to transmit the final report until one of them is confirmed.

2.5.1. Chainlink data feed implementation

Here we have the code used by Chainlink in order to retrieve the price of a certain token. In this case we want to know the price of Ethereum over US Dollars.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";

contract PriceConsumerV3 {
    AggregatorV3Interface internal priceFeed;

    /**
     * Network: Goerli
     * Aggregator: ETH/USD
     * Address: 0xD4a33860578De61DBAbDc8BFdb98FD742fA7028e
     */
    constructor() {
        priceFeed = AggregatorV3Interface(
            0xD4a33860578De61DBAbDc8BFdb98FD742fA7028e
        );
    }

    /**
     * Returns the latest price
     */
    function getLatestPrice() public view returns (int) {
        (
            ,
            /*uint80 roundID*/ int price /*uint startedAt*/ /*uint timeStamp*/
/*uint80 answeredInRound*/,
            ,
            ,

        ) = priceFeed.latestRoundData();
        return price;
    }
}
```

By analysing the source code, we are able to notice that an aggregator is used. Multiple separate Chainlink oracle operators update each data feed. On-chain data is aggregated by the AccessControlledOffchainAggregator.

A decentralized oracle network updates each data feed. Each Oracle operator gets compensated for data publication. The number of oracles involved in each feed varies. The data feed aggregator contract must receive responses from a minimum number of Oracles for an update to occur; otherwise, the most recent answer will not be updated. During an aggregation round, each oracle in the collection broadcasts data. A smart contract validates and aggregates that data, forming the feed's most recent and reliable answer.

Data Feeds are an example of a decentralized Oracle network, and they consist of the following elements:

- **Consumer contract:** A consumer contract is any contract that consumes aggregated data from Chainlink Data Feeds. Consumer contracts must refer to the appropriate AggregatorV3Interface contract and invoke one of the available functions. Data feeds can also be consumed by off-chain applications. To learn more, see the Javascript and Python example code on the Using Data Feeds page.
- **Proxy contract:** On-chain proxies that point to the aggregator for a specific data stream are known as proxy contracts. Using proxies allows the underlying aggregator to be upgraded without affecting consuming contracts.
- **Aggregator contract:** The contract that receives periodic data updates from the Oracle network is known as an aggregator. Aggregators keep aggregated data on-chain so that users can access it and act on it in the same transaction. Only when the Deviation Threshold or Heartbeat Threshold triggers an update during an aggregation round do aggregators receive updates from the Oracle network. When the first condition is met, the data is updated.

Why we need to go to that address? That is a question that will be answered in the next sub-chapter.

Following the address and inserting it on Etherscan we're able to analyse the contract in detail. It will not be posted here for space reasons; we'll take a look only on certain lines.

```
function latestRoundData()
    public
    view
    checkAccess()
    override
    returns (
      uint80 roundId,
      int256 answer,
      uint256 startedAt,
      uint256 updatedAt,
      uint80 answeredInRound
    )
  {
    return super.latestRoundData();
  }
```

Contract source code 3.2.1

This is the contract function that is used to retrieve the price feed. Customers are recommended to review the updatedAt and answeredInRound return values to ensure that they are receiving new data from the most recent round. It should be noted that different underlying AggregatorV3Interface implementations have slightly different meanings for certain of the return values. Consumers should establish which implementations they intend to receive data from and ensure that they can handle all of them appropriately. [10]

- roundId is the round id from the aggregator for which the data was retrieved combined with a phase to ensure that round IDs get larger as time moves forward.
- Answer is the answer for the given round.
- StartedAt is the timestamp when the round was started.
- updatedAt is the timestamp when the round last was updated.

- answeredInRound is the round ID of the round in which the answer was computed.
- note that answer and updatedAt may change between queries.

```
function latestRoundData()
    public
    view
    virtual
    override
    returns (
      uint80 roundId,
      int256 answer,
      uint256 startedAt,
      uint256 updatedAt,
      uint80 answeredInRound
    )
  {
    Phase memory current = currentPhase; // cache storage reads

    (
      uint80 roundId,
      int256 answer,
      uint256 startedAt,
      uint256 updatedAt,
      uint80 ansIn
    ) = current.aggregator.latestRoundData();

      return addPhaseIds(roundId, answer, startedAt, updatedAt, ansIn,
current.id);
  }
```

An on-chain mapping of assets to feeds is provided by the Chainlink Feed Registry, in the proposed case the lastestRoundData function, is the one who will interact with the Feed Registry. Without knowing the feed contract addresses, it enables you to immediately access Chainlink data feeds from asset addresses. They make it possible for smart contracts to call a single contract and receive the most recent pricing for an asset.

Solidity's Storage and Memory keywords are equivalent to a computer's hard disk and RAM. Memory in Solidity, like RAM, is a temporary location to store data, whereas Storage stores data between function calls. The Solidity Smart Contract can utilize any amount of memory during execution, but once the execution is complete, the memory is wiped clean for the next execution. In contrast to Storage, which is durable, each execution of the Smart contract has access to the data previously saved on the storage area.

3.2.2.1 Blockchain’s addresses

In Ethereum and Solidity, an address if of 20 byte value size (160 bits or 40 hex characters). It corresponds to the last 20 bytes of the Keccak-256 hash of the public key. An address is always pre-fixed with 0x as it is represented in hexadecimal format (base 16 notation) (defined explicitly). On the blockchain, an address functions as someone’s identity. The address's connection to a wallet address, smart contract, or transaction hash is determined by this information. Addresses come in two varieties: Contract Addresses and Externally Owned Addresses, which is essentially your wallet address. A wallet address, sometimes referred to as an externally owned address (EOA), is a public account that contains your wealth and can only be accessed by private key pairs. In essence, the Ethereum address is the "public" address you'd need to send money to someone else using the Ethereum network. Accordingly, the money won't show up in the recipient's wallet address if the network is on a different network. Make that the address is compatible with the fund and the network being used to send the fund. Additionally, you need the address's private key to access any money stored there. It is recommended to handle the private key with caution as it can be used to access all the funds in an address.

The term "contract address" describes the location of a group of executable pieces of code on the Ethereum blockchain. When a transaction with related input data (a contract interaction) is made to a contract address, several functions are carried out. When a contract is deployed to the Ethereum Blockchain, the contract address

is typically produced. Both Externally Owned and Contract Addresses include 42 hexadecimal characters, which is a common format.

3.3 Differences between Provable and Chainlink

The Oracle Provable and Chainlink protocols use slightly different approaches to provide external data to smart contracts on the blockchain, so their code may vary in some parts. Here are some of the main differences in the code of the two protocols:

- **Architecture**: Oracle Provable uses a server-based structure to collect and deliver data to smart contracts. This means that the protocol code includes components such as test servers, which verify the veracity of the data provided, and signature servers, which create the digital signatures used to guarantee the integrity of the data. In contrast, Chainlink uses a network of geographically distributed oracle nodes to collect and verify external data. The protocol code therefore includes components such as oracle nodes and data verification scripts.
- **Programming languages**: Oracle Provable was mainly developed in Go, a programming language designed for the creation of distributed and scalable systems. Chainlink, on the other hand, was mainly developed in Solidity, a programming language specifically for the creation of smart contracts on the Ethereum blockchain.
- **Data Management**: Oracle Provable uses a centralised database to store the data provided to oracles. Chainlink, on the other hand, uses a blockchain-based approach to store the data provided by oracles. This means that the data is stored on the blockchain in a decentralised manner, making it accessible to any oracle node in the network.

Here's a code example showing how the Provable oracle procides a proof regarding data integrity to a smart contract:

```
pragma solidity ^0.4.22;
import "https://github.com/oraclesorg/oracle-provable-
contracts/provableAPI_0.5.sol";

contract MyContract {
    using Provable for *;

  // Calling Oracle Provable test server to receive the price of a stock
    function getStockPrice(string symbol) public
      view returns (uint price) {
        bytes32 query = abi.encodePacked("URL", symbol);
        (price) = Provable.query("URL", query);
    }

    // verifies the signature for data integrity
    function verifyData() public view {
        bytes32 queryId = Provable.lastQueryId();
        assert(Provable.proof(queryId).isValid());
    }
}
```

getStockPrice: This function is called by a smart caontract to obtain the price of a stock from a specific stock symbol. The function uses the Oracle Provable API to send a request to the test server with the stock symbol as a parameter. The function returns the share price as a result.

verifyData: This function is called by a smart contract to verify the integrity of the data provided by Oracle Provable's test server. The function uses Oracle Provable's API to retrieve the last request sent to the test server and verify that its proof of data integrity is valid. If the data integrity test is valid, the function terminates without generating an error. Otherwise, the function generates an error to indicate that the data has not been verified.

verifySignature: This function is used to verify the digital signature of the test server on the data provided by the server. The function takes the data and digital signature as input and uses the ecrecover function to calculate the signer's address from the digital signature. If the address of the signer matches the address of the test server, the function returns true, otherwise it returns false.

On the other hand this is the solution used by Chainlink when returning data to a smart contract:

```
pragma solidity ^0.4.22;
import "https://github.com/smartcontractkit/chainlink/blob/stable/evm-
contracts/src/v0.6/ChainlinkClient.sol";

contract MyContract {
    ChainlinkClient public client;

    // The Chainlink oracle's node wants to obtain the price of a stock
    function getStockPrice(string symbol) public
      view returns (uint price) {
        bytes32 requestId = chainlinkRequest(symbol);
        price = oracleResponse(requestId);
    }

    // This funcion sends a request to the Chainlink oracle's node
    function chainlinkRequest(string symbol) public
      view returns (bytes32 requestId) {
        requestId = client.createRequest(
            "URL",
            abi.encodePacked(symbol)
        );
        client.setFulfillmentPermission(requestId, oracle, true);
    }

    // This function receives the Chainlink oracle's node response
    function oracleResponse(bytes32 requestId) public
      view returns (uint price) {

     //  Verifies that the response was signed by the authorized oracle node
        assert(client.getFulfillmentSignature(requestId) == oracleSignature);
        // Decodes the oracle node response
        (price) = abi.decode(client.getFulfillment(requestId));
    }
}
```

Here are some of the main differences between the two pieces of code:

Architecture: Oracle Provable's code includes a call to the *Provable.query* function to obtain data from the proof server, while Chainlink's code includes a call to the *client.createRequest* function to send a request to the oracle node.

Oracle's Provable code also includes a call to the *Provable.proof* function to verify the signature of the data integrity proof, while Chainlink's code includes a call to the *client.getFulfillmentSignature* function to verify the signature of the response from the oracle node.

Programming languages: Oracle Provable's code is written in Solidity, while Chainlink's code is written in Solidity. However, Oracle Provable's code includes an import of an Oracle Provable contract library written in Go, while Chainlink's code includes an import of a Chainlink contract library written in Solidity.

Data Management: Oracle Provable's code does not include any logic for managing the data provided by the test server, since this data is stored in a centralised database. Chainlink's code, on the other hand, includes logic to receive and decode the data provided by the oracle node, since this data is stored on the blockchain in a decentralised manner.

Here is a detailed description of the functions in the Chainlink code:

getStockPrice: This function is called by a smart contract to obtain the price of a stock from a specific stock symbol. The function sends a request to Chainlink's oracle node with the stock symbol as a parameter and returns the stock price as a result.

chainlinkRequest: This function is called from the getStockPrice function to send a request to Chainlink's oracle node. The function uses the Chainlink API to create a request and set the oracle node as the authorised recipient of the request. The function returns the request ID as a result.

oracleResponse: This function is called from the getStockPrice function to receive the response from Chainlink's oracle node. The function uses Chainlink's API to verify that the response has been signed by the authorised oracle node and, if so, decodes the response to extract the stock price. The function returns the share price as a result.

getFulfillmentSignature: This function is used to verify the digital signature of the oracle node on the response provided by the node. The function takes as input the

request ID, the payment due to the oracle node, the client callback function and the response from the oracle node. The function uses the *isSigned* function to verify that the response has been signed by the oracle node and, if the verification is positive, sends the response to the client using the callback function.

Some of the vulnerabilities that could be exploited by a hacker to attack Oracle Provable code are:

- **Test server attack**: a hacker could attempt to compromise the Oracle Provable test server in order to alter the data provided to the oracles or prevent access to the data.
- **Digital signature attack**: A hacker could attempt to forge the digital signature used to verify the integrity of data provided by Oracle Provable, which could render the data provided by the test server untrustworthy.
- **Attack on the oracle network**: a hacker could attempt to compromise one or more Oracle Provable oracle nodes to alter the data provided to the oracles or prevent access to the data.

Some of the vulnerabilities that could be exploited by a hacker to attack Chainlink's codes are:

- **Oracle node attack**: a hacker could attempt to compromise Chainlink's oracle node to alter the data provided to oracles or prevent access to the data.
- **Digital signature attack**: An attacker could attempt to forge the digital signature used by the Chainlink oracle node to verify the integrity of the response, which could render the data provided by the oracle node untrustworthy.
- **Attack on the oracle network**: A hacker could attempt to compromise one or more of Chainlink's oracle nodes to alter the data provided to oracles or prevent access to data.

The following code is an example of Provable oracle calling the function getStockPrice:

```
pragma solidity ^0.6.6;

import "https://github.com/provable-things/
ethereum-api/sol/ProvableAPI.sol";

contract StockPriceOracle is ProvableAPI {
    // The address of the Oracle contract
    address public oracle;

    constructor() public {
        oracle = provable_newOracle("YOUR_ORACLE_NAME");
    }

    // The function that calls the Oracle
    function getStockPrice(string memory _symbol) public
      view returns (uint) {

        bytes memory query =
            abi.encodePacked("{'symbol':'" + _symbol + "'}");
        (bool success, bytes memory result) =
            oracle.call("getStockPrice", query);
        require(success, "Error calling the Oracle");

        // Parse the result
        (uint price,) = abi.decode(result, (uint));
        return price;
    }
}
```

In this smart contract, the function *provable_newOracle* is used to create a new Oracle Provable oracle with the specified name. Next, the function *getStockPrice* is defined, which takes the stock symbol as input and uses the oracle.call function to invoke the *getStockPrice* function of the Oracle Provable oracle. The *oracle.call* function returns the result of the call to the oracle in the form of bytes, so the *abi.decode* function must be used to extract the price from the bytes and return it as the result of the *getStockPrice* function.

The following code is an example of Chainlink oracle calling the function getStockPrice:

```
pragma solidity ^0.6.6;

import "https://github.com/smartcontractkit/chainlink/solidity/
contracts/ChainlinkClient.sol";

contract StockPriceOracle is ChainlinkClient {
    // The address of the Oracle contract
    address public oracle;

   constructor() public {
        oracle = CHAINLINK_ORACLE_ADDRESS;
    }

    // The function that calls the Oracle
    function getStockPrice(string memory _symbol) public
            view returns (uint) {
        // The function that will be called by the Oracle
        function callback(bytes memory _response) public
            view returns (uint) {
            // Parse the response from the Oracle
            (uint price,) = abi.decode(_response, (uint));
            return price;
        }

        // The request to be sent to the Oracle
        Request memory req =
        buildChainlinkRequest(
            oracle,
            this,
            callbackId,
            bytes4(keccak256(abi.encodePacked("getStockPrice(string)"))),
            oracle.funcSelector("getStockPrice(string)"),
            abi.encodePacked(_symbol)
        );

        // Send the request
        sendChainlinkRequestTo(oracle, req, ORACLE_PAYMENT);
        return 0;
    }
}
```

In this smart contract, the constant CHAINLINK_ORACLE_ADDRESS is used to specify the address of the Chainlink oracle to be used. Next, the *getStockPrice* function is defined which takes the stock symbol as input and uses the *buildChainlinkRequest* function to create a request to be sent to the Chainlink

oracle. The *buildChainlinkRequest* function takes as input the address of the oracle, the address of the contract sending the request, the ID of the callback function, the selector of the *getStockPrice* function and the action symbol encoded in bytes.

The callback function is defined as a function within the contract and is used to extract the price from the oracle's response and return it as the result of the *getStockPrice* function. Finally, the *sendChainlinkRequestTo* function is used to send the request to the oracle and the contract returns the value 0 while waiting for the oracle's response.

3.4 Conclusions

In conclusion, it's clear that both Provable and Chainlink are powerful and versatile oracle systems, each with their own strengths and weaknesses. Chainlink excels in terms of security, as it is a fully decentralized oracle system, which means that it is less susceptible to security problems. However, it is also worth noting that it is less efficient in terms of response times when compared to Provable.

On the other hand, Provable boasts faster response times, making it a great option for real-time data feeds and applications. However, it should be noted that it is not fully decentralized, which may be a concern for some users. Ultimately, the choice between Provable and Chainlink will depend on the specific needs of the application and its users. If security is the top priority, then Chainlink may be the better choice, but if faster response times are needed, then Provable may be the better option.

The implementation of oracle consensus, which ensures that multiple sources are providing the same data, should be considered as well. This can improve the reliability of the data provided by oracles. Chainlink provides a decentralized oracle network that allows multiple independent node operators to provide the same data, making it more resistant to manipulation. Provable, being a centralized oracle, does not have this feature but it could still implement a similar approach of having multiple sources for the same data.

In any case, both oracles have been widely adopted and implemented in various use cases, showing their robustness and reliability in the blockchain ecosystem. The development of new oracles and their integration with smart contract platforms will make the deployment of decentralized applications more efficient, providing a wide range of possibilities for the future.

Chapter 4

New oracle proposals

4.1. Optimistic oracles

The emergence of blockchain technology has opened up new possibilities for decentralized systems, but it has also created new challenges, particularly when it comes to incorporating real-world data into these systems. The oracle problem, which refers to the challenge of securely and accurately connecting blockchains with real-world data, has been a topic of much research and development in recent years.

One of the most promising solutions to the oracle problem is the use of Optimistic Oracles, a new type of oracle technology that leverages the strengths of blockchain technology to provide secure, transparent, and efficient integration of real-world data into decentralized systems. Optimistic Oracles work by allowing transactions to be temporarily accepted based on real-world data, with a dispute resolution mechanism in place to correct any invalid outcomes. [11]

In this chapter, we will provide a technical overview of Optimistic Oracles, including how they work, their advantages over traditional oracle solutions, and their limitations. We will also delve into the key components of Optimistic Oracles, including data provision, transaction execution, dispute resolution, and correction of invalid outcomes. Finally, we will provide a detailed analysis of the cost, security, and reliability of Optimistic Oracles, and discuss the future directions for this exciting new technology.

The aim of this chapter is to provide a comprehensive and in-depth understanding of Optimistic Oracles, and to explore the potential for this technology to revolutionize the way real-world data is integrated into decentralized systems. Whether you are a researcher, developer, or blockchain enthusiast, this chapter is

an essential resource for anyone interested in the future of oracle technology and the role it will play in shaping the decentralized future.

4.1.1. Optimistic oracles and their objective

Optimistic Oracles are a type of oracle that aim to provide fast, efficient, and secure data integration in blockchain networks. The objective of Optimistic Oracles is to allow for real-time data integration while preserving the security and trustworthiness of the data being incorporated into the blockchain.

Optimistic Oracles work by allowing transactions to proceed optimistically, meaning that they are executed immediately and the outcome is temporarily accepted as valid. If a dispute arises or the outcome is found to be invalid, the transaction is rolled back and the correct outcome is established through a dispute resolution mechanism. This allows for fast transaction processing times, while still ensuring that the final outcome is accurate and trustworthy.

The use of Optimistic Oracles can be particularly useful in decentralized finance (DeFi) applications where fast, secure, and accurate data integration is critical for the functioning of financial instruments such as loans, stablecoins, and derivatives.

4.1.2. How do they work?

Optimistic Oracles work by allowing transactions to proceed optimistically, meaning that they are executed immediately and the outcome is temporarily accepted as valid. The key components of Optimistic Oracles are:

Data Providers: These entities provide real-world data to the blockchain network. They can be individuals, organizations, or smart contracts.

Oracle Contracts: These smart contracts receive the data from the data providers and incorporate it into the blockchain network.

Dispute Resolution Mechanisms: These mechanisms allow for the resolution of disputes that may arise between parties involved in a transaction. Dispute

resolution mechanisms may include consensus-based methods, such as voting, or more formal methods, such as arbitration.

The process of incorporating data into the blockchain network using Optimistic Oracles typically works as follows:

- The data provider provides real-world data to the oracle contract.
- The oracle contract executes the transaction based on the received data and temporarily accepts the outcome as valid.
- The transaction is broadcast to the network and its validity is temporarily accepted.
- If a dispute arises, a dispute resolution mechanism is triggered, and a consensus is reached on the correct outcome.
- If the outcome is found to be invalid, the transaction is rolled back and the correct outcome is established.

This process allows for fast, efficient, and secure data integration while preserving the security and trustworthiness of the data being incorporated into the blockchain network.

4.1.3. Analysis of the points made.

When it comes to incorporating real-world data into the blockchain network using Optimistic Oracles, the following steps occur:

- **Data Provision**: The data provider, which can be an individual, organization, or smart contract, provides the relevant real-world data to the oracle contract.
- **Transaction Execution**: The oracle contract receives the data from the data provider and executes a transaction based on the received data. The outcome of the transaction is temporarily accepted as valid.
- **Broadcast to the Network**: The transaction is broadcast to the network, and its validity is temporarily accepted by the network participants.

- **Dispute Resolution**: In the event of a dispute, the dispute resolution mechanism is triggered. The dispute resolution mechanism can vary depending on the specific implementation, but it is typically designed to reach a consensus on the correct outcome.
- **Correction of Invalid Outcomes**: If the outcome of the transaction is found to be invalid, the transaction is rolled back, and the correct outcome is established. This process helps to maintain the security and trustworthiness of the data incorporated into the blockchain network.

Overall, this process allows for fast and efficient data integration while preserving the security and trustworthiness of the data. The specific implementation of the Optimistic Oracle, including the dispute resolution mechanism, will determine the level of security and reliability of the solution.

4.1.3.1. Broadcast to the network

In the context of Optimistic Oracles, broadcasting a transaction to the network refers to the process of propagating the transaction across the network of nodes that make up the blockchain. This process is a crucial step in the integration of real-world data into the blockchain, as it allows the network to temporarily accept the validity of the transaction based on the real-world data.

The specific details of the broadcast process will depend on the blockchain platform being used, but in general, it can be broken down into the following steps:

- Serialization: The transaction is transformed into a binary format, known as a serialized transaction, which can be transmitted across the network.
- Node Discovery: The node that created the transaction must discover other nodes in the network to which it can transmit the serialized transaction. This process can be accomplished using a variety of techniques, including broadcasting the transaction directly to known nodes, using a peer-to-peer discovery protocol, or using a centralized node discovery service.

- Transmission: The serialized transaction is transmitted to the discovered nodes. The nodes that receive the transaction will validate it and add it to their local copy of the blockchain.
- Propagation: The nodes that have received the transaction will propagate it further by transmitting it to other nodes in the network. This process continues until the transaction has been broadcast to a sufficient number of nodes, ensuring that it has been widely propagated across the network.
- Validation: Upon receipt of the transaction, each node will validate it to ensure that it is valid and conforms to the rules of the blockchain platform. This process involves checking the authenticity of the transaction and the data provided by the Optimistic Oracle.
- Confirmation: Once the transaction has been validated, it is temporarily confirmed, and its validity is accepted by the network. The transaction will be included in the next block that is added to the blockchain.

This broadcast process allows the network to temporarily accept the validity of the transaction and its associated real-world data, allowing for fast and efficient data integration. However, if a dispute arises, the dispute resolution mechanism will be triggered to reach a consensus on the correct outcome, ensuring the security and trustworthiness of the data in the blockchain network.

4.1.3.2. Dispute resolution

Dispute resolution is a critical aspect of Optimistic Oracles, as it provides a mechanism for resolving disputes that arise from differences in opinions regarding the validity of the real-world data incorporated into the blockchain network. The dispute resolution process is triggered when a disagreement arises regarding the validity of the transaction based on the real-world data.

The specific details of the dispute resolution process will vary depending on the implementation of the Optimistic Oracle, but it typically involves the following steps:

- Dispute Initiation: The dispute is initiated by a network participant, who may be a node operator, a smart contract, or a user. The participant will provide evidence to support their claim that the transaction outcome is incorrect.
- Evidence Collection: The dispute resolution mechanism will collect evidence from all parties involved in the dispute. This may include data from the oracle contract, the real-world data provider, and other relevant sources.
- Consensus Building: The dispute resolution mechanism will use the collected evidence to build consensus among the network participants regarding the correct outcome of the transaction. This may involve voting, staking, or other consensus-building mechanisms.
- Resolution: Based on the consensus reached, the dispute resolution mechanism will resolve the dispute by determining the correct outcome of the transaction. This may involve modifying the transaction, rolling it back, or taking other corrective action.
- Correction: Once the dispute has been resolved, the oracle contract will implement the correction, updating the blockchain with the correct outcome of the transaction.

The goal of the dispute resolution process is to ensure the security and trustworthiness of the real-world data incorporated into the blockchain network. The specific dispute resolution mechanism used will determine the level of security and reliability of the Optimistic Oracle solution. In general, it is important to choose a mechanism that is transparent, fair, and efficient, and that provides strong incentives for network participants to act in the best interests of the network.

4.1.3.3. Correction of invalid outcomes

The correction of invalid outcomes in Optimistic Oracles is done through the dispute resolution process. When a dispute arises regarding the validity of the transaction based on the real-world data, the dispute resolution mechanism is

triggered. If the outcome of the transaction is found to be invalid, the dispute resolution mechanism will determine the correct outcome and make the necessary corrections.

The specific correction process will depend on the implementation of the Optimistic Oracle and the dispute resolution mechanism used, but it typically involves the following steps:

- Rollback: The first step in correcting an invalid outcome is to roll back the original transaction to its pre-execution state. This involves removing the transaction from the blockchain and restoring the state of the network to what it was before the transaction was executed.
- Updating the oracle contract: The next step is to update the oracle contract with the correct outcome of the transaction. This may involve modifying the data stored in the oracle contract, adding new data, or updating the logic of the contract.
- Re-execution of the transaction: After the oracle contract has been updated, the transaction can be re-executed using the correct outcome. This will ensure that the blockchain reflects the correct state of the network and that the real-world data is accurately represented on the blockchain.
- Confirmation: Finally, the correction will be confirmed by the network participants, who will validate the corrected transaction and add it to the blockchain.

It is important to note that the correction process should be designed to be transparent, fair, and efficient. The dispute resolution mechanism should provide strong incentives for network participants to act in the best interests of the network and should be designed to minimize the impact of invalid outcomes on the network. Additionally, the correction process should be designed to minimize the risk of further disputes and to ensure the long-term security and reliability of the Optimistic Oracle solution.

4.1.4. Optimistic oracles security

Optimistic Oracles can be secure, but the level of security depends on the specific implementation and the design of the dispute resolution mechanism. In general, Optimistic Oracles aim to provide fast and efficient data integration while preserving the security and trustworthiness of the data.

Compared to traditional oracles like Chainlink and Provable, Optimistic Oracles can be faster since transactions can be executed immediately and temporarily accepted as valid, allowing for real-time data integration. However, the security of Optimistic Oracles may be lower compared to traditional oracles, as the dispute resolution mechanism may not be as robust or secure as consensus-based methods used by traditional oracles.

Ultimately, whether someone should use Optimistic Oracles instead of traditional oracles depends on the specific requirements of the application and the trade-off between speed, cost, and security. If a fast, efficient, and secure data integration solution is needed, Optimistic Oracles may be a good option. However, if a higher level of security is required, traditional oracles such as Chainlink or Provable may be a better choice.

4.1.5. Cost of Optimistic oracles

The cost of using Optimistic Oracles can vary depending on the specific implementation and the cost of the dispute resolution mechanism. In general, the cost of using Optimistic Oracles may be lower compared to traditional oracles such as Chainlink and Provable, since they can allow for faster transaction processing times and can reduce the need for consensus-based methods.

However, the exact cost will depend on various factors, such as the scale of the application, the complexity of the dispute resolution mechanism, and the cost of gas on the blockchain network. Additionally, some Optimistic Oracles may require

a higher upfront cost to set up and maintain the oracle contract and the dispute resolution mechanism.

Ultimately, the cost of using Optimistic Oracles should be evaluated in the context of the specific requirements of the application and the trade-off between cost, speed, and security. If cost is a major concern, Optimistic Oracles may be a good option, but it is important to consider all factors before making a decision.

4.1.6. Conclusion for optimistic oracles

In conclusion, Optimistic Oracles represent a promising new solution to the oracle problem in the blockchain world. By leveraging the strengths of blockchain technology, Optimistic Oracles provide secure, transparent and efficient integration of real-world data into decentralized systems.

In this chapter, we have provided a technical overview of Optimistic Oracles, exploring the key components of this technology and the advantages it offers over traditional oracle solutions. We have also analysed the cost, security and reliability of Optimistic Oracles, and discussed the future directions for this exciting new technology. It is clear that Optimistic Oracles have the potential to play a significant role in shaping the future of decentralized systems and the integration of real-world data into these systems. However, as with any new technology, there are challenges and limitations that must be addressed. Further research and development will be needed to fully realize the potential of Optimistic Oracles and to ensure their long-term viability and success.

This chapter provides a comprehensive understanding of Optimistic Oracles and their place in the blockchain world. Whether you are a researcher, developer or blockchain enthusiast, it is our hope that this chapter has provided valuable insights and a foundation for continued exploration of this exciting new technology.

4.2. Introduction to layer 2 zero-knowledge rollups

In recent years, the scalability and security of blockchain systems have become major challenges in the development and widespread adoption of decentralized applications. To address these challenges, various scaling solutions have been proposed, including Layer 2 (L2) Zero-Knowledge (ZK) rollups. This chapter provides an overview of L2 ZK rollups and their role in the scaling of blockchain systems. The chapter will explain how L2 ZK rollups work, including the off-chain aggregation of transactions, the generation of zero-knowledge proofs, and the on-chain validation of transactions. The chapter will also compare L2 ZK rollups with other scaling solutions, such as traditional oracles and optimistic oracles, in terms of security and speed. The goal of this chapter is to provide a comprehensive understanding of L2 ZK rollups and their role in the scaling of blockchain systems, and to highlight the trade-offs and considerations involved in the use of this technology. [12]

Layer 2 zero-knowledge rollups

L2 (Layer 2) ZK (Zero-Knowledge) rollups refer to a type of scaling solution for blockchain systems that leverages Zero-Knowledge proofs to increase the transaction processing capacity of a blockchain network while maintaining the security guarantees of a blockchain.

L2 ZK rollups work by batching multiple transactions into a single, larger transaction, which is then processed on the blockchain using a Zero-Knowledge proof to ensure that the batch of transactions is valid. By using Zero-Knowledge proofs, L2 ZK rollups can greatly increase the transaction processing capacity of a blockchain network, as the batch of transactions can be processed as a single, efficient transaction, rather than as many individual transactions.

In addition to increasing transaction processing capacity, L2 ZK rollups also offer improved security, as the Zero-Knowledge proof provides a cryptographic

guarantee that the transactions in the batch are valid and cannot be manipulated. This makes L2 ZK rollups a promising solution for the scaling of blockchain systems and the integration of decentralized applications into these systems.

4.2.1. How do they work?

L2 ZK rollups work by aggregating multiple transactions into a single, larger transaction, which is then processed on the blockchain. The process can be broken down into the following steps:

- **Off-chain aggregation**: Multiple transactions are grouped together into a single batch and processed off-chain. This allows for greater efficiency and faster processing times, as the batch of transactions can be processed as a single unit.
- **Zero-Knowledge proof generation**: A Zero-Knowledge proof is generated to demonstrate that the batch of transactions is valid. This proof is a compact representation of the batch of transactions that can be verified by the blockchain without revealing the details of the individual transactions.
- **On-chain validation**: The batch of transactions and the Zero-Knowledge proof are submitted to the blockchain for validation. The blockchain uses the Zero-Knowledge proof to verify that the batch of transactions is valid and that no transactions in the batch have been altered or manipulated.
- **Transaction execution**: If the batch of transactions is found to be valid, it is executed on the blockchain, and the individual transactions in the batch are processed as a single, larger transaction. This greatly increases the transaction processing capacity of the blockchain, as many transactions can be processed in a single block.

Overall, L2 ZK rollups work by leveraging the strengths of Zero-Knowledge proofs to increase the transaction processing capacity of a blockchain network while maintaining the security guarantees of a blockchain. By aggregating multiple transactions into a single, efficiently processed unit, L2 ZK rollups provide a

promising solution for the scaling of blockchain systems and the integration of decentralized applications into these systems.

4.2.1.1. Off-chain aggregation

Off-chain aggregation refers to the process of grouping multiple transactions into a single unit and processing them outside of the blockchain. This is typically done by a smart contract, known as an aggregator contract, which is responsible for collecting, organizing and validating the transactions.

The off-chain aggregation process involves the following steps:

- Collection: Transactions are collected by the aggregator contract, which listens for incoming transactions and adds them to its pool.
- Organization: The transactions are organized into a batch, and the aggregator contract checks each transaction to ensure it is valid and conforms to the established rules and conditions.
- Validation: The aggregator contract validates each transaction in the batch to ensure it is valid and that the conditions specified in the contract have been met. This includes checking for any errors or inconsistencies in the transaction data and ensuring that the transaction is authorized.
- Grouping: The validated transactions are grouped into a single, larger transaction, known as a batch. This batch of transactions is then processed as a single unit, allowing for greater efficiency and faster processing times.

By aggregating multiple transactions into a single unit, off-chain aggregation allows for faster and more efficient processing of transactions, reducing the load on the blockchain and enabling greater scalability.

4.2.1.2. zero knowledge proof

Zero-Knowledge proofs (ZKPs) are a powerful tool in the world of cryptography that allow for secure and private verification of information. In the

context of L2 ZK rollups, they are used to efficiently verify the validity of a large number of transactions without revealing any sensitive information. This is achieved through a combination of complex mathematical computations and cryptographic techniques.

The process of generating a Zero-Knowledge proof starts with a prover, which can be a smart contract or a centralized party, creating a proof that demonstrates the validity of a batch of transactions. The prover performs a series of computations, including hash functions and public key cryptography, to create a compact representation of the proof. This representation includes a commitment to the inputs and outputs of the transactions in the batch, as well as a proof of the validity of the computations performed by the prover.

Once the proof is generated, it is then verified by a verifier, such as a node on the blockchain network. The verifier performs the same computations as the prover and checks the consistency of the proof to ensure that the transactions in the batch are valid. If the proof is verified, the validity of the transactions is established without revealing any additional information.

In this way, ZKPs are a key tool for improving the security, efficiency, and privacy of transactions on the blockchain. They allow for the efficient verification of a large number of transactions, while keeping sensitive information private and secure.

4.2.1.3. On-chain validation

On-chain validation refers to the process of verifying the validity of transactions directly on the blockchain. In this process, the validation is performed by nodes on the network, which are responsible for checking the validity of the transactions and adding them to the blockchain.

On-chain validation works by having each node on the network validate each transaction before it is added to the blockchain. This validation process involves checking that the transaction is formatted correctly, that it follows the rules of the

blockchain's consensus algorithm, and that it does not conflict with other transactions in the blockchain.

If a node determines that a transaction is invalid, it will reject it and the transaction will not be added to the blockchain. On the other hand, if the transaction is determined to be valid, the node will broadcast it to the rest of the network, and it will be added to the blockchain once a sufficient number of nodes have confirmed its validity.

The on-chain validation process is a crucial aspect of maintaining the security and integrity of the blockchain. It helps to ensure that only valid transactions are added to the blockchain, and that the state of the blockchain remains consistent. By relying on a decentralized network of nodes for validation, the blockchain is protected from malicious actors who might attempt to manipulate the network or add invalid transactions.

4.2.1.4. Transaction execution

Transaction execution in a blockchain refers to the process of executing the instructions specified in a transaction and updating the state of the blockchain accordingly. The following is an overview of the transaction execution process:

- Creation of a transaction: A user creates a transaction that specifies the instructions to be executed, such as transferring funds from one account to another or modifying the state of a smart contract.
- Broadcasting the transaction: The user broadcasts the transaction to the network, where it is propagated to all nodes on the network.
- Validation of the transaction: The nodes on the network validate the transaction to ensure that it is properly formatted, follows the rules of the blockchain's consensus algorithm, and does not conflict with other transactions in the blockchain.

- Adding the transaction to a block: If the transaction is determined to be valid, it is added to a block, which is a collection of validated transactions. The block is then broadcast to the network and added to the blockchain.
- Execution of the instructions: Once the block containing the transaction is added to the blockchain, the instructions specified in the transaction are executed. For example, if the transaction is a transfer of funds, the balance of the sender's account will be reduced and the balance of the recipient's account will be increased.
- Updating the state of the blockchain: The state of the blockchain is updated to reflect the execution of the transaction. For example, the new account balances will be recorded in the blockchain.

This process is repeated for each transaction that is added to the blockchain, allowing the state of the blockchain to be updated in real-time as transactions are executed and validated. The execution of transactions on the blockchain is secure and transparent, as it is based on a decentralized network of nodes and validated through consensus algorithms.

4.2.2. ZK-rollups security

L2 ZK rollups are considered to be a secure solution for the oracle problem in the blockchain world. However, the level of security of any solution will depend on the specific implementation and use case.

In comparison to traditional oracles such as Chainlink or Provable, L2 ZK rollups offer several advantages:

- Scalability: L2 ZK rollups allow for off-chain aggregation of transactions, which can significantly increase the overall scalability of the network. This can result in faster transaction processing times and lower fees.
- Security: By using zero-knowledge proofs, L2 ZK rollups provide a secure way to validate the validity of transactions without revealing the underlying data. This enhances the overall security of the network and reduces the risk of tampering or manipulation of data.

- Cost: L2 ZK rollups can be more cost-effective than traditional oracles, as they allow for the aggregation of multiple transactions into a single transaction. This can result in lower fees compared to the cost of executing multiple transactions on-chain.

It's important to note that while L2 ZK rollups offer these advantages, they also have their own set of challenges and limitations. As with any technology, the decision to use L2 ZK rollups should be based on a thorough evaluation of the specific use case and requirements.

L2 Zk Rollups are a new technology in the blockchain space that aims to address scalability, privacy and security concerns in blockchain networks. They operate by aggregating multiple transactions off-chain, applying zero-knowledge proofs to verify the validity of the transactions, and then executing them on-chain. In comparison to traditional oracles like Chainlink or Provable, L2 Zero-Knowledge Rollups offer benefits such as faster transactions and improved scalability due to off-chain aggregation. However, there are some disadvantages to consider as well, such as the requirement for specialized technical knowledge and the need for more complex infrastructure to implement L2 ZK Rollups. Additionally, the security and privacy of L2 ZK Rollups may not be on par with that of traditional oracles, depending on the specifics of each implementation. It's important to carefully evaluate the trade-offs before choosing which technology to use.

ZK rollups are considered secure, as they leverage the security guarantees of the underlying blockchain and use zero-knowledge proofs to provide a cryptographic guarantee that the transactions in the batch are valid and cannot be manipulated. This provides a high level of security, as the transactions are validated on-chain and are recorded on the blockchain as a permanent part of its history.

Compared to traditional oracles like Chainlink and Provable, ZK rollups offer some benefits in terms of security and scalability. ZK rollups provide a more secure and efficient solution for off-chain transactions, as the transactions are executed on-chain and are recorded on the blockchain, making them a permanent part of its history. In contrast, traditional oracles rely on a trusted third party to

provide data to the blockchain, which can introduce potential security vulnerabilities.

In terms of speed, ZK rollups can be faster than traditional oracles, as the transactions are executed on-chain as a single, larger transaction, rather than as many individual transactions. This allows for increased scalability and efficiency, as the batch of transactions can be processed more quickly than many individual transactions.

Overall, the use of ZK rollups as a scaling solution depends on the specific use case and the requirements of the user. ZK rollups provide increased security and scalability compared to traditional oracles, but they may also have some limitations and trade-offs, such as higher costs and the need for a more complex setup. Ultimately, the decision to use ZK rollups or another solution depends on the specific needs and requirements of the user.

4.2.3. Differences with optimistic oracles

Optimistic Oracles and ZK rollups are both scaling solutions for blockchain systems, but they have some key differences in terms of security and speed.

In terms of security, ZK rollups are considered more secure as they use zero-knowledge proofs to provide a cryptographic guarantee that the transactions in the batch are valid and cannot be manipulated. In contrast, Optimistic Oracles rely on an optimistic assumption that the data provided by the oracle is correct and can be verified on-chain. If the oracle provides incorrect data, this can lead to security vulnerabilities in the system.

In terms of speed, Optimistic Oracles can be faster than ZK rollups in certain scenarios, as they can allow for transactions to be executed off-chain and then verified on-chain. This allows for faster transaction processing and increased scalability, as the transactions can be executed more quickly than if they were processed on-chain. However, in scenarios where the volume of transactions is

high, ZK rollups may offer better performance, as the batching of transactions allows for more efficient processing.

Ultimately, the choice between Optimistic Oracles and ZK rollups depends on the specific use case and the requirements of the user. Both solutions offer benefits and trade-offs in terms of security and speed and the decision to use one or the other should be based on the specific needs and requirements of the user.

4.2.4. Conclusions

In conclusion, L2 ZK rollups represent a promising solution for the scaling of blockchain systems and the integration of decentralized applications. By leveraging zero-knowledge proofs, L2 ZK rollups provide increased transaction processing capacity while maintaining the security guarantees of a blockchain. The technology works by batching transactions off-chain, generating zero-knowledge proofs, and validating transactions on-chain. However, it is important to consider the trade-offs and limitations of L2 ZK rollups when deciding whether to implement this technology in a given system. By comparing L2 ZK rollups to other scaling solutions, such as traditional oracles and optimistic oracles, the advantages and disadvantages of each can be better understood. Ultimately, the choice between these technologies depends on the specific requirements and goals of the system in question. This chapter has aimed to provide a comprehensive overview of L2 ZK rollups and to offer insights into their potential use and implementation in blockchain systems.

Conclusion

After a thorough examination of oracles in blockchain technology, we can conclude that while they offer numerous benefits in terms of efficiency, automation and transparency, they also introduce significant security risks. The Oracle Problem, which refers to the challenge of ensuring the reliability of information from oracles, remains a significant obstacle to the widespread adoption of blockchain technology. As the use cases for blockchain technology continue to expand, the importance of finding effective solutions to the Oracle Problem becomes even more critical.

Dopo aver analizzato nel dettaglio I differenti tipi di oracoli e il loro ruolo nella blockchain technology possiamo dedurre che

After analysing in detail the different types of oracles e their role in blockchain technology, we can deduce that there is no one-size-fits-all solution to the Oracle Problem. Different approaches have their own advantages and limitations, and the choice of solution will depend on the specific use case and the level of security required. For example, if a high level of security is essential, Chainlink may be the best option due to its decentralized approach, even though it may be more expensive. On the other hand, if cost is a significant concern, Provable may be a better choice, although this may come at a potential trade-off in terms of security.

Looking to the future, there is no doubt that blockchain technology will continue to evolve, and with it, the role of oracles. The increasing interest in the use of blockchain technology in various industries, such as finance, supply chain management and insurance, means that finding effective solutions to the Oracle Problem is becoming even more critical. The development of new consensus algorithms, smart contracts and oracle networks is likely to play a significant role in addressing the security concerns associated with oracles in blockchain technology.

In conclusion, this thesis has provided a comprehensive understanding of oracles in blockchain technology, examining their concepts, applications and security issues. By exploring different approaches to address the Oracle Problem, we have identified potential solutions that can be implemented to improve the security and reliability of information provided by oracles. As blockchain technology continues to mature, it is likely that the importance of oracles will only increase, making it essential to continue exploring the challenges and opportunities associated with their use in blockchain systems.

Bibliography

[1] Chainlink, “What is blockchain,” 03 08 2022. [Online]. Available: https://blog.chain.link/what-is-blockchain/.

[2] Crypto.com, “consensus mechanisms in blockchain,” Crypto.com, 13 05 2022. [Online]. Available: https://crypto.com/university/consensus-mechanisms-in-blockchain. [Accessed 06 10 2022].

[3] D. L. T. S. J. K. K. E. A. B. H. L. F. K. W. W. Darren Lau, “DeFi for Beginners,” in *DeFi for Beginners*, 2021.

[4] Cointelegraph, “What is a smart contract, and how does it work?,” Cointelegraph, [Online]. Available: https://cointelegraph.com/learn/what-are-smart-contracts-a-beginners-guide-to-automated-agreements. [Accessed 02 08 2022].

[5] Chainlink, “what is the blockchain oracle problem?,” Chainlink, 27 08 2020. [Online]. Available: https://blog.chain.link/what-is-the-blockchain-oracle-problem/. [Accessed 03 08 2022].

[6] Chainlink, “Hybrid smart contracts,” 18 05 2021. [Online]. Available: https://blog.chain.link/hybrid-smart-contracts-explained/. [Accessed 03 08 2022].

[7] Chainlink, “What is a blockchain oracle?,” 14 09 2021. [Online]. Available: https://chain.link/education/blockchain-oracles. [Accessed 10 08 2022].

[8] CyStack Editor, “DeFi Security: Situation and Solution – Examples of DeFi Attacks Included,” CyStack.net, 16 08 2022. [Online]. Available: https://cystack.net/blog/defi-security-best-practices. [Accessed 10 10 2022].

[9] Provable, “Security Deep Dive,” Provable, [Online]. Available: http://docs.provable.xyz/#security-deep-dive. [Accessed 21 12 2022].

[10] Chainlink, “Feed Registry,” [Online]. Available: https://docs.chain.link/data-feeds/feed-registry. [Accessed 21 12 2022].

[11] H. Lambur, “Oracles are vital to DeFi. Here's how they are evolving,” 14 03 2022. [Online]. Available: https://forkast.news/oracles-are-vital-to-defi-heres-how-they-are-evolving/. [Accessed 02 02 2023].

[12] Empiric Network, 15 06 2022. [Online]. Available: https://medium.com/@EmpiricNetwork/empiric-network-the-zk-native-oracle-dcff3c36658a. [Accessed 01 02 2023].

[13] Chainlink, 24 01 2022. [Online]. Available: https://blog.chain.link/what-is-blockchain/.