Chapter 9
Data Authenticity

Ken Huang

Data authenticity is a critical security concern for web3 applications that utilize real-world data with on-chain smart contracts. When the data upon which a smart contract relies is inaccurate or tampered with, it can have serious unintended consequences. This is an especially daunting challenge since smart contracts operate automatically and cannot independently verify the accuracy or integrity of the data they receive. As a result, security vulnerabilities and loss of trust in the system are likely outcomes.

Fortunately, several companies, including Chainlink, Band Protocol, UMA, Nest Protocol, and API3, have stepped up to address this issue by providing decentralized data feeders or decentralized oracles to smart contracts.

These decentralized oracles provide a secure and reliable way to connect smart contracts to off-chain data sources, ensuring that the data they receive is accurate and trustworthy. By utilizing a decentralized network of data providers, these oracles ensure that data is verified and validated through a consensus mechanism, reducing the risk of manipulation or malicious attacks.

This chapter discusses different types of data oracles; examples of data oracle providers; oracle use cases in DeFi, NFT, Insurance, Enterprise, prediction market, and even sustainability; oracle design considerations, and security attacks on oracles and the countermeasures.

K. Huang (✉)
DistributedApps.AI, Fairfax, VA, USA
e-mail: Ken@Distributedapps.ai

K. Huang et al. (eds.), *A Comprehensive Guide for Web3 Security*, Future of Business and Finance, https://doi.org/10.1007/978-3-031-39288-7_9

9.1 Types of Blockchain Oracles

Different types of oracles are needed for smart contracts because the requirements for external data and computation can vary widely between different contracts. For example, some contracts may need real-time data from external sources, while others may be able to work with delayed data. Some contracts may require data from multiple sources, while others may only need data from one specific source.

Additionally, the level of security required for the data can also differ between contracts. Some contracts may be designed to handle sensitive financial or personal data and therefore require a higher level of security, while others may not require such stringent measures.

To meet these diverse requirements, different types of oracles are designed to handle specific tasks. For example, some oracles are specialized in fetching data from a specific type of source, while others are designed to perform computations on the data before delivering it. These different types of oracles allow smart contracts to be tailored to meet the specific needs of each use case, which is essential for their functionality and success.

9.1.1 Input Oracles

The input oracle is the most well-known type of oracle in use today, and it plays a crucial role in enabling smart contracts to interact with the real world. As its name suggests, the input oracle fetches data from off-chain sources and delivers it to the blockchain network for consumption by smart contracts.

In Fig. 9.1, the input data oracle is represented by the rectangle labeled "Input Data Oracle," while the smart contract is represented by the rectangle labeled "Smart Contract." The arrow indicates that the input data oracle sends input data to the smart contract.

One of the most popular applications of input oracles is in powering Chainlink Price Feeds. These feeds provide DeFi (Decentralized Finance) smart contracts with on-chain access to financial market data, such as cryptocurrency prices or exchange rates. This data is crucial for the functioning of many DeFi applications, such as lending platforms, stablecoins, and derivatives trading platforms.

For example, consider a DeFi lending platform that uses an input oracle to retrieve real-time cryptocurrency prices. When a user deposits their crypto assets as collateral, the platform's smart contract uses the price data from the oracle to calculate the value of the collateral and determine the maximum loan amount that can be

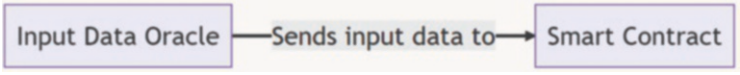

Fig. 9.1 Input Data Oracle

issued. If the price of the collateral drops below a certain threshold, the smart contract will automatically trigger a liquidation event to repay the loan and protect the lender's funds.

In this scenario, the input oracle serves as a trusted source of financial market data, enabling the smart contract to operate securely and autonomously. This is just one of many examples of how input oracles are used to bring real-world data into the blockchain and power the next generation of decentralized applications.

9.1.2 Output Oracles

Output oracles serve as the opposite of input oracles, as they allow smart contracts to communicate with the real-world (off-chain) systems and trigger them to perform specific actions. These actions can range from simple tasks, such as sending a notification, to more complex processes, such as executing a financial transaction or controlling a physical device.

Output oracles are used **after** smart contract execution to provide data and to trigger real-world events while input oracles are used **before** smart contract execution to provide real-world data to rigger smart contract execution.

In Fig. 9.2, the smart contract is represented by the rectangle labeled "Smart Contract," the output data oracle is represented by the rectangle labeled "Output Data Oracle," and the real-world application is represented by the rectangle labeled "Real-World Application." The arrows indicate that the smart contract sends the execution result to the output data oracle, and the output data oracle sends the output data to the real-world application.

For example, consider a smart contract that manages car rentals on a blockchain network. Once a user makes an on-chain payment for a rental, the smart contract can send a command to an output oracle, which then pings an IoT (Internet of Things) system to unlock the car door. This type of output oracle provides a secure and automated way for the smart contract to communicate with the off-chain world and execute the rental process.

Another example is a smart contract that manages payments for a cloud storage service. The smart contract can use an output oracle to send a command to the storage provider, instructing them to store the specified data. This eliminates the need for manual intervention, reduces the risk of human error, and provides a more efficient and secure way to manage the storage service.

Output oracles play an important role in enabling smart contracts to interact with the real-world systems and perform a wide range of tasks. By providing a secure and automated way to communicate with off-chain systems, output oracles help to

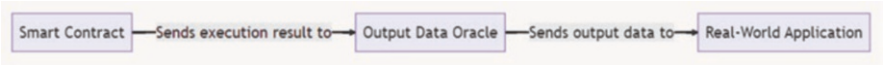

Fig. 9.2 Output Data Oracle

bring the benefits of blockchain technology to a wider range of use cases and industries.

9.1.3 Cross-Chain Oracles

Cross-chain oracles are a type of oracle that enable communication and transfer of information and assets between different blockchain networks. They play a crucial role in enabling interoperability across different blockchains, making it possible to move data and assets between them.

In Fig. 9.3, there are two chains—Chain A and Chain B, represented by the rectangles labeled "Chain A" and "Chain B," respectively. The cross-chain data oracle is represented by the rectangle labeled "Cross-Chain Data Oracle," and the real-world application is represented by the rectangle labeled "Real-World Application." The arrows indicate that Chain A sends a request to the cross-chain data oracle, which fetches the data from Chain B and sends it to both Chain A and the real-world application.

For example, consider a decentralized exchange that operates on a blockchain network. The exchange uses a cross-chain oracle to retrieve data from another blockchain network, such as the current price of an asset. This allows the exchange to offer trading in assets that are native to different blockchains, making it possible for users to trade assets across different networks.

Another example is a decentralized finance (DeFi) platform that operates on one blockchain network, but allows users to use assets from another blockchain network. The DeFi platform uses a cross-chain oracle to bridge assets between the two networks, making it possible for users to use their assets in the DeFi platform, even though they were originally issued on a different blockchain network.

Cross-chain oracles play a critical role in promoting blockchain interoperability and helping to advance the decentralized finance ecosystem. By enabling seamless communication and transfer of information and assets between different blockchains, cross-chain oracles help to break down barriers and create a more connected and interoperable blockchain landscape.

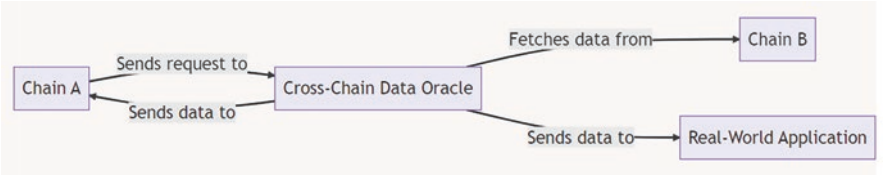

Fig. 9.3 Cross-Chain Oracles

9.1.4 Compute-Enabled Oracles

Compute-enabled oracles are an emerging type of oracle that is gaining popularity in smart contract applications. These oracles use secure off-chain computation to offer decentralized services that may be impractical or impossible to perform on-chain due to technical, legal, or financial limitations.

In Fig. 9.4, the smart contract is represented by the rectangle labeled "Smart Contract," the Chainlink node is represented by the rectangle labeled "Chainlink Node," the compute-enabled oracle is represented by the rectangle labeled "Compute-Enabled Oracle," and the external API is represented by the rectangle labeled "External API." The arrows indicate the flow of data and computation. The smart contract requests data from the Chainlink node, which routes the request to the compute-enabled oracle. The compute-enabled oracle uses data from the external API to perform computation and sends the result back to the Chainlink node, which then sends the result to the smart contract.

For instance, Chainlink Automation uses compute-enabled oracles to trigger smart contracts execution based on predefined real-world events, automating complex processes and eliminating the need for manual intervention. This approach enhances efficiency and security in smart contract execution.

Another example is the use of zero-knowledge proofs, a form of secure computation that enables privacy-preserving computations. Compute-enabled oracles can perform these proofs off-chain, providing smart contracts with a secure way to access private data without exposing it to the public.

Additionally, compute-enabled oracles can run verifiable randomness functions, which are critical for decentralized applications such as gaming and lotteries. These oracles offer a tamper-proof and provably fair source of randomness to smart contracts, ensuring that they operate in a transparent and equitable manner.

9.1.5 Other Types of Oracles

In addition to oracles listed above in this chapter, there are other types of oracles that can provide valuable services to smart contracts. These include:

- Trusted Computing Environment (TEE) is a technology that is increasingly used to protect sensitive data. For high-end sensors or servers, the TEE can be used to

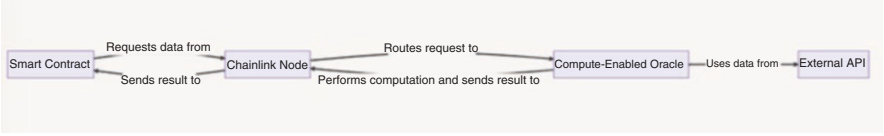

Fig. 9.4 Compute-Enabled Oracle

digitally sign collected data, ensuring that data cannot be tampered with during transmission. The TEE physically protects the private key used for signing, making the key invalid without the physical subject. This guarantees that only authorized parties can access the data and that any unauthorized changes to the data will be unsuccessful. Other similar technologies that can be used for protection include Secure Element Physical UnclonableFunction (PUF) and other technologies. Estonia is an excellent example of how TEE can be used to protect authenticity of voters record. The country allows citizens to vote electronically using their national ID cards, which contain a TEE that ensures the private key is physically protected and cannot be used without the physical ID card. This makes it virtually impossible for anyone to hack the voting system or tamper with votes during transmission (Lohrmann, 2020).

- On-chain asset tracking is another way to ensure data authenticity. This technology is used for assets with unique physical properties that can be measured. For instance, Everledger uses the unique physical properties of diamonds to create a chain of custody for precious stones, preventing the circulation of fake diamonds (Williams & Kaplan, 2017). This technology ensures that the data about the asset is stored in a distributed ledger that is tamper-proof, making it difficult for fraudsters to manipulate the data.
- Artificial intelligence and machine learning can also be used to increase the authenticity of data. These technologies can be used to identify and filter out fake data, making it easier to verify the authenticity of uplinked data. By analyzing large volumes of data, these technologies can detect patterns and anomalies, helping to identify fake data.
- Video surveillance and other security controls are another important component of data authenticity. These technologies can be used to monitor assets and provide additional security to prevent fraud or theft. For example, video surveillance can be used to monitor the movement of assets and identify any suspicious activities. Other security controls can be used to restrict access to data, making it more difficult for unauthorized parties to gain access.

9.2 Examples of Data Oracle Providers

There are several ways to ensure the authenticity of data provided by a data oracle, including:

Cryptographic techniques: Data oracles can use cryptographic techniques, such as digital signatures and hash functions, to ensure the integrity and authenticity of the data being provided.

Decentralization: Data oracles can be implemented as decentralized networks, which can help to ensure the authenticity of the data by making it harder for a single entity to control or manipulate the data.

Governance: Data oracles can use decentralized governance models to ensure the authenticity of the data by aligning the interests of oracle operators with those of the network.

Reputation: Data oracles can use reputation systems to ensure the authenticity of the data by incentivizing oracle operators to provide accurate and reliable data.

Ensuring the authenticity of data provided by a data oracle is important for the integrity and trustworthiness of a blockchain application, as it helps to ensure that the smart contract is executing correctly and producing reliable results.

This section highlights several oracle providers which used cryptographic techniques, decentralization, governance, and reputation to improve data authenticity.

Chainlink
Chainlink is a decentralized oracle network that allows smart contracts to access external data and off-chain resources. It uses a network of independent oracle nodes to fetch data from external sources and feed it into the blockchain via secure and transparent mechanisms. Chainlink has been widely adopted by a number of blockchain projects and has partnerships with major companies and organizations.

Band Protocol
Band Protocol is a decentralized oracle platform that allows developers to build their own oracles and use them to access external data and information. It uses a decentralized governance model to ensure the integrity and trustworthiness of the oracles on the platform. Band Protocol has been used in a number of decentralized applications (DApps) and has partnerships with major companies and organizations.

UMA
UMA (Universal Market Access) is a decentralized financial contracts platform that uses data oracles to provide real-time pricing data to smart contracts. It allows users to create and trade synthetic assets that track the value of real-world assets, such as stocks, commodities, and currencies. UMA uses a decentralized governance model to ensure the integrity and reliability of its oracles.

Nest Protocol
Nest Protocol is a decentralized oracle platform that allows developers to build their own oracles and use them to access external data and information. It uses a decentralized governance model and a system of incentives to ensure the integrity and trustworthiness of the oracles on the platform. Nest Protocol has been used in a number of decentralized applications and has partnerships with major companies and organizations.

Table 9.1 compares each oracle provider based on publicly available sources. Cryptographic techniques used by these projects may vary depending on specific use cases and implementations. The degree of decentralization is subjective and may change over time. Governance models can also change as projects evolve. Reputation technologies can be either on-chain or off-chain.

Each of the data oracles listed in the table has a unique approach to solving the data authenticity problem in web3 applications. For example, Chainlink utilizes

threshold signatures as a consensus mechanism to ensure that data is verified by a network of trusted data providers.

Band Protocol, on the other hand, utilizes a delegated proof of stake consensus mechanism, which allows token holders to elect validators to provide data.

UMA takes a different approach, relying on economic incentives to encourage data providers to submit accurate data.

Nest Protocol utilizes a unique nesting consensus mechanism that combines multiple sources of data to arrive at a consensus, while API3 uses a community-driven governance approach to ensure the accuracy and reliability of data.

Overall, the architecture, consensus mechanisms, and incentives used by each data oracle differ significantly, and developers must carefully evaluate the pros and cons of each solution when selecting a data oracle for their project. By doing so, they can ensure that their smart contracts have access to reliable and trustworthy data, improving the security and functionality of their web3 applications.

The following provides further analysis of the benefits and drawbacks of each data oracle approach, with a focus on scalability, reliability, and cost-effectiveness:

1. Chainlink:
   - Benefits:
     - Decentralized architecture provides high security and reliability.
     - Threshold signatures consensus mechanism ensures data accuracy and tamper-proofing.
     - LINK tokens incentivize data providers to submit accurate data.
   - Drawbacks:

Table 9.1 Data oracle providers comparison

Project	Cryptographic Techniques	Degree of Decentralization	Governance	Reputation
Chainlink	Elliptic curve cryptography, hash functions, digital signatures	Highly decentralized	Decentralized governance	On-chain reputation
Band protocol	Multi-party computation, threshold signatures, hash functions	Moderately decentralized	DAO governance	On-chain reputation
UMA	Zero knowledge proofs, hash functions, digital signatures	Moderately decentralized	Tokenholder governance	On-chain reputation
Nest protocol	Elliptic curve cryptography, hash functions, digital signatures	Moderately decentralized	DAO governance	On-chain reputation
API3	Hash functions, digital signatures, threshold signatures	Highly decentralized	DAO governance	Off-chain reputation

  - Requires a large number of trusted data providers to achieve a high level of decentralization, which may limit scalability.
  - Cost may be a factor for small-scale projects, as the use of LINK tokens may be expensive.

2. Band Protocol:
   - Benefits:
     - Delegated proof of stake consensus mechanism provides fast and efficient verification of data.
     - BAND tokens incentivize data providers to submit accurate data.
     - Decentralized architecture provides high security and reliability.
   - Drawbacks:
     - The use of delegated proof of stake may limit decentralization and reduce security.
     - The need for token holders to elect validators may limit scalability.
3. UMA:
   - Benefits:
     - Economic incentives incentivize data providers to submit accurate data.
     - Decentralized architecture provides high security and reliability.
     - Allows for customizable data feeds.
   - Drawbacks:
     - Economic incentives may be expensive and may not be sustainable in the long term.
     - The need for users to hold UMA tokens may limit adoption and scalability.
4. Nest Protocol:
   - Benefits:
     - The nesting consensus mechanism provides a high level of accuracy and reliability.
     - NEST tokens incentivize data providers to submit accurate data.
     - Hybrid architecture provides scalability and decentralization.
   - Drawbacks:
     - The nesting consensus mechanism may be complex and difficult to implement.
     - The need for NEST tokens may limit adoption and scalability.
5. API3:
   - Benefits:

  - Community-driven governance provides high transparency and accountability.
  - Hybrid architecture provides scalability and decentralization.
  - API3 tokens incentivize data providers to submit accurate data.

- Drawbacks:

  - Community-driven governance may be slow and cumbersome.
  - The need for API3 tokens may limit adoption and scalability.

In addition to the oracle providers listed above, there are other oracle providers used in various business uses cases:

Witnet
Witnet is a decentralized oracle network where nodes earn or lose reputation based on their correct or incorrect fulfillment of data requests, as determined by a consensus algorithm. Nodes are randomly chosen for jobs and mining blocks based on their network reputation, making majority attacks more difficult. Reputation is constantly redistributed at each block to prevent centralization and exit scams, using a demurrage function. Witnet is a separate blockchain that provides decentralized oracle services via bridge nodes, offering a scalable solution with reduced on-chain operation fees and the ability to fix critical vulnerabilities as a last resort. More information can be found at https://witnet.io or in their whitepaper.

Oraclize
Oraclize is a London-based cybersecurity company offering a centralized solution to blockchain oracles. While it is available on multiple blockchain platforms (Bitcoin, Ethereum, Monax, Rootstock, Corda, and private networks), the majority of their customers are working on Ethereum.

Their approach is to leverage all TEE environment providers to minimize vulnerability. This is what they call sandboxing. Oraclize leverages the products of IT providers and manufacturers (including Amazon's EC2, Google's SafetyNet, Qualcomm's QSEE, Ledger's Nano S and Intel's SGX) as key components of its own core service (the Oraclize technology). They are physically grouped within a unique environment and leveraged together: Oraclize has designed ad-hoc custom applications as well as a software layer connection for all of TEEs to make them interoperable. By collecting the data from multiple TEEs, even if one technology were to be compromised by a vulnerability such as Spectre for Intel's SGX, the overall aggregation of value would still ignore the compromised data point (assuming the vulnerability was architecture specific, and not a generic one hitting all processors).

To achieve distributed trust and the integrity of their data, Oraclize has been relying on TLS-Notary to digitally sign TLS data from https websites. This comes at cost: Oraclize can in theory only deliver data as shown on the website with no post-processing off-chain, but this already covers many use cases. The main risk here remains that if too many data sources are compromised, there is no way to prevent

wrong data from being propagated, but this risk is also present in the more "decentralized" solutions.

Town Crier
Town Crier acts as a bridge between smart contracts on any blockchain and https-enabled websites with TLS layer handling handshakes for secure communication to deliver source-authenticated data. The approach taken is different to TLS-notary (security at software level only), allowing for more customizable data relaying.

The data is collected by nodes running on Intel's SGX (security at both software and hardware level). This authenticated data feed is delivered from enclave to the blockchain, solely relying on the SGX protection to testify the node is indeed running the software as expected.

To protect confidentiality, messages are only decrypted inside the Trusted Execution Environment's enclave, which can thus be used not only for safe data transfer but also for ingesting encrypted user credentials (e.g., private API). In addition, custom requests are supported for potentially multiple web-scraping target.

Their approach to tackle single points-of-failure is to aggregate both data source and data oracles on multiple SGX platforms. The software has proven to be relatively scalable with throughputs of 15–65 transactions/sec.

9.3 Oracle Use Cases

As the world becomes increasingly digital, the need for reliable and accurate data has never been more important. In the realm of Web3, this is where data oracles come in. The importance of data oracles cannot be overstated, as they are essential for ensuring trust, security, and transparency in various industries, including DeFi, NFTs, insurance, enterprise supply chain management, prediction markets, and sustainability initiatives. In this section, we will explore the different use cases of data oracles in each of these areas and discuss some examples. By the end of this section, you will have a better understanding of the critical role data oracles play in the Web3 ecosystem and how they can be leveraged to transform various industries.

9.3.1 Decentralized Finance (DeFi)

A large portion of the decentralized finance (DeFi) ecosystem requires oracles to access financial data about assets and markets. For example, decentralized lending protocol uses price oracles to determine users' borrowing capacity and check if users' positions are undercollateralized and subject to liquidation. Similarly, synthetic asset platforms use price oracles to peg the value of tokens to real-world assets and automated market makers (AMMs) use price oracles to help concentrate liquidity at the current market price to improve capital efficiency.

There are numerous examples of successful implementations of oracle data in DeFi. Here are a few notable examples:

Aave: Aave is a lending platform that uses oracles to determine the value of collateral. This allows the platform to adjust loan-to-value ratios based on changes in the value of the collateral.

Synthetix: Synthetix is a platform that allows users to trade synthetic assets. Oracles are used to provide price feeds for these assets, allowing users to trade them with confidence.

There are a variety of technologies and formulas used in the collection and analysis of oracle data in DeFi. Here are a few examples:

Linear interpolation: Linear interpolation is a technique used to estimate a value between two known values. This is commonly used in price feeds, where the price of an asset may not be available at a specific point in time.

Weighted average: Weighted average is a technique used to calculate an average based on the importance or weight of each data point. This is commonly used in price feeds, where the price of an asset may be based on multiple exchanges or data sources.

Time-weighted average price (TWAP): TWAP is a formula used to calculate the average price of an asset over a specified period of time. This is commonly used in price feeds to provide a more accurate representation of the true value of an asset.

Oracle data plays a critical role in the DeFi ecosystem, providing reliable, accurate data for a variety of applications. By following best practices and using trusted, decentralized oracles, DeFi platforms can ensure that.

9.3.2 Dynamic NFTs and Gaming

Oracles enable dynamic NFTs (Non-Fungible Tokens that can change in appearance, value, or distribution based on external events like the time of day or the weather). Additionally, compute oracles are used to generate verifiable randomness that projects then use to assign randomized traits to NFTs or to select random lucky winners in high-demand NFT drops. On-chain gaming applications also use verifiable randomness to create more engaging and unpredictable gameplay experiences like the appearance of random loot boxes or randomized matchmaking during a tournament. The following are example projects:

Dynamic NFTs: An example of a project that uses oracles for dynamic NFTs is Ether Cards, which is a platform that allows users to create and collect customizable NFT cards with dynamic traits. Ether Cards uses Chainlink VRF (Verifiable Randomness Function) to generate random traits for the cards, such as discounts, access rights, royalties, etc. Ether Cards also uses Chainlink API calls to enable the cards to change based on external events, such as sports scores, crypto prices, social media trends, etc.

Verifiable randomness for NFT traits: The “CryptoPunks” NFT collection is an example of NFTs that use verifiable randomness generated by oracles. Each

CryptoPunk has a randomized set of traits (e.g., punk hair, accessories, facial features) that were assigned using a verifiable randomness function. This ensures that no two CryptoPunks are identical and creates scarcity and uniqueness within the collection.

Randomized gameplay experiences: The on-chain game "Axie Infinity" uses oracles to generate verifiable randomness for various gameplay elements. For example, the appearance of random loot boxes that contain valuable in-game items is determined by an oracle-generated random number. Additionally, randomized matchmaking during tournaments ensures that players are paired up in a fair and unpredictable way, creating a more engaging and challenging gameplay experience.

9.3.3 Insurance

Insurance smart contracts use input oracles to verify the occurrence of insurable events during claims processing, opening up access to physical sensors, web APIs, satellite imagery, and legal data. Output oracles can also provide insurance smart contracts with a way to make payouts on claims using other blockchains or traditional payment networks.

The following is the examples of oracle use in insurance.

Arbol: Arbol uses IoT sensors to monitor weather conditions, such as temperature, rainfall, and wind speed, in real time. The data collected by the sensors is then fed into smart contracts, which automatically trigger payouts to policyholders if certain predefined conditions are met. For example, if the temperature in a certain area drops below a certain threshold, a smart contract could automatically initiate a payout to farmers who have taken out weather-related insurance policies (Arbol, 2021).

Etherisc: Etherisc uses oracles to access external data sources, such as flight information APIs, to determine whether a policyholder's flight has been delayed or cancelled. When a delay or cancellation occurs, the smart contract automatically triggers a payout to the policyholder based on the terms of their insurance policy (Cuenca, 2022).

AIG and Standard Chartered: AIG, IBM, and Standard Chartered completed a pilot project in 2017 that used smart contracts and oracles to automate trade finance transactions, including insurance coverage. The project involved using smart contracts to automatically trigger insurance payouts when certain conditions were met, such as when goods were delivered to a certain location. Oracles were used to provide external data, such as shipping information, to the smart contracts (BusinessWire, 2017).

9.3.4 Enterprise Supply Chain Management

Supply chain management is the process of planning, coordinating, and executing the flow of materials, products, and information from suppliers to customers in an efficient and effective way. Data oracles are used to connect blockchain-based smart contracts to external data sources, such as sensors, APIs, databases, or other blockchains. Data oracles can provide information such as temperature, location, quality, quantity, and status of goods along the supply chain to smart contracts. This can enable automation, verification, and optimization of supply chain processes. Some examples of companies using data oracles in their supply chain management are ADNOC, De Beers, Walmart, Zara, and Everledger (Sharma, 2022).

Abu Dhabi National Oil Company (ADNOC): ADNOC has partnered with IBM to use blockchain to track, validate, and execute transactions across its oil and gas value chain. The blockchain platform uses smart contracts and IoT devices to automate the accounting and reconciliation processes, reduce operational costs, and increase transparency.

De Beers: De Beers has developed a blockchain platform called Tracr to trace the provenance and quality of diamonds from mine to retail. The platform uses smart contracts and digital certificates to verify the authenticity and ethical sourcing of diamonds, as well as to prevent fraud and theft.

Walmart: Walmart has collaborated with IBM and other partners to use blockchain to improve food safety and traceability. The blockchain platform uses smart contracts and IoT sensors to record data such as temperature, location, and expiration date of food products along the supply chain. This enables Walmart to quickly identify and recall contaminated products, as well as to reduce waste and spoilage.

Zara: Zara has implemented a blockchain solution called Loomia to enhance its fast fashion supply chain. The solution uses smart contracts and RFID tags to track the movement of garments from production to distribution to retail. This helps Zara optimize its inventory management, reduce costs, and increase customer satisfaction.

Everledger: Everledger is a company that uses blockchain to create digital passports for wine bottles. The digital passports contain information such as grape variety, vintage, origin, quality, and ownership history of each bottle. The blockchain platform uses smart contracts and NFC tags to verify the authenticity and provenance of wine bottles, as well as to prevent counterfeiting and fraud.

9.3.5 Prediction Markets

Decentralized prediction markets are platforms that allow users to create and trade on the outcome of events, such as political elections, sports games, or even the weather. These markets operate using a decentralized network of participants, who

buy and sell shares in various outcomes based on their predictions of the event's outcome.

To participate in a prediction market, users must first buy shares in the possible outcomes of the event. For example, in a prediction market for a political election, users might buy shares in the candidates they think will win. The price of these shares fluctuates based on supply and demand, with the price of the winning outcome eventually settling at 1 and the price of the losing outcome settling at 0.

In order to determine the final outcome of the event and settle the market, prediction markets rely on oracles. Oracles are trusted sources of off-chain data that are used to determine the outcome of the event. For example, in a prediction market for a sports game, an oracle might be used to provide the final score of the game.

Augur and Gnosis, two popular decentralized prediction market platforms, each use their own unique oracle systems.

Augur uses a decentralized network of oracles, which are selected by Augur token holders based on their reputation and accuracy in reporting outcomes. Once an outcome is determined, the oracles report the result to the Augur smart contract, which automatically settles the market and pays out winnings to the users who correctly predicted the outcome.

Gnosis, on the other hand, uses a centralized oracle system called the Gnosis Olympia Oracle. The Olympia Oracle is a consortium of trusted data providers who are responsible for reporting the outcome of events to the Gnosis platform. Once an event is resolved, the Gnosis smart contract uses the Olympia Oracle's data to determine the final outcome and automatically settle the market.

9.3.6 Sustainability

In recent years, there has been a growing interest in using blockchain technology and smart contracts to support environmental initiatives, such as carbon offsetting and sustainable land management. Oracles play a critical role in these efforts by providing smart contracts with environmental data from a variety of sources, including sensors, satellite imagery, and advanced machine learning computation. This data can then be used to automatically trigger rewards, verify the effectiveness of environmental projects, and support the creation of new forms of carbon credits. Here are some examples of companies and projects that are using oracles to support these efforts:

Supplying Smart Contracts with Environmental Data

Ocean Protocol: Ocean Protocol is a decentralized data exchange that allows individuals and organizations to share and monetize their data. The platform uses oracles to securely connect data sources, such as sensors and satellite imagery, with smart contracts. This allows the data to be used for a variety of applications, including tracking environmental metrics and supporting sustainable practices.

ClimateTrade: ClimateTrade is a carbon offset marketplace that uses blockchain and smart contracts to automate the purchase and verification of carbon offsets. The platform uses oracles to gather environmental data, such as carbon emissions data from industrial sources, and feed it into smart contracts. These smart contracts then automatically initiate the purchase and transfer of carbon offsets to buyers, providing a more transparent and efficient way to offset carbon emissions.

ClimateCHECK: ClimateCHECK is a software platform that helps companies track, measure, and report their carbon emissions. The platform uses oracles to gather environmental data from a variety of sources, such as energy usage data and transportation data, and feed it into smart contracts. These smart contracts then automatically calculate the company's carbon emissions and provide recommendations for reducing them.

Supporting New Forms of Carbon Credits

Nori: Nori is a marketplace for carbon removal credits that uses blockchain and smart contracts to automate the purchase and verification of carbon removal services. The platform uses oracles to gather environmental data, such as soil carbon levels and tree growth rates, from a network of trusted data providers. This data is then used to verify the effectiveness of carbon removal projects and provide a more transparent and trustworthy way to buy and sell carbon removal credits.

9.4 Oracle Design Considerations

Oracles play a critical role in the blockchain ecosystem by providing off-chain data to smart contracts. However, designing an oracle requires careful consideration of several factors to ensure the accuracy, reliability, and security of the data provided to the smart contract. The design considerations for oracles include correctness, availability, incentive compatibility, data quality, security, scalability, and cost transparency. Correctness involves ensuring the authenticity and integrity of the data provided to the smart contract to prevent state changes based on invalid information. Availability is essential to enable smart contracts to access off-chain data without delay or interruption. Incentive compatibility incentivizes off-chain data providers to submit accurate information, enabling rewards or penalties based on information quality. Data quality is crucial to ensuring the accuracy of the smart contract's execution, and the oracle should source data from multiple trusted sources and verify the accuracy of the data. Security measures such as encryption, authentication, and authorization must be integrated into the oracle design to prevent malicious actors from manipulating the data. Scalability is also critical, and the oracle must be designed to handle a high number of requests per second and manage data storage and retrieval efficiently. Lastly, the cost of using the oracle should be transparent and cost-effective. In this article, we will explore each of these design considerations in detail and discuss best practices for designing oracles that meet the needs of the blockchain ecosystem.

Correctness: An oracle should provide valid off-chain data that does not cause smart contracts to execute state changes based on invalid information. This requires the authenticity and integrity of data, meaning that it was obtained from the correct source and not tampered with before being sent on-chain. For example, Certified Origins is using the Oracle Blockchain Platform (OBP) to verify the authenticity of its Italian extra virgin olive oil and to streamline billing and purchase orders (Shaw, 2020).

Availability: An oracle should ensure that smart contracts can access off-chain data without delay or interruption, enabling them to execute actions and trigger state changes promptly. For example, Chainlink oracle networks aggregate data from a number of decentralized ChainLink nodes to remove any single point of failure in the delivery of data to the blockchain. This provides strong guarantees to users around the availability, accuracy, and the tamper-proof nature of sourcing off-chain data and delivering it on-chain.

Incentive Compatibility: An oracle should incentivize off-chain data providers to submit accurate information to smart contracts. This involves attributability and accountability, allowing for the correlation of external information to its provider and the bonding of data providers to the information they provide, enabling rewards or penalties based on information quality.

Data Quality: The quality of data provided by the oracle must be high to ensure the accuracy of the smart contract's execution. The oracle should source data from multiple trusted sources and have a mechanism to verify the accuracy of the data.

Security: The oracle must be secure to prevent malicious actors from manipulating the data provided to the smart contract. The oracle should be designed with security features such as encryption, authentication, and authorization.

Scalability: The oracle must be scalable to handle the large volume of data required by Web3 applications. The oracle should be designed to handle a high number of requests per second and have a mechanism to manage data storage and retrieval efficiently.

Cost: The cost of using the oracle should be considered when designing the oracle. The oracle should be cost-effective, and the cost of using the oracle should be transparent to the users.

In addition to the previously mentioned oracle design patterns, there are two other commonly used patterns that depend on the specific use case: request-response and publish-subscribe. The request-response pattern allows client contracts to request specific data that may be too large to store on the blockchain, while the publish-subscribe pattern provides a data feed that can be read by multiple smart contracts.

Publish-subscribe Oracles

An oracle service based on a publish-subscribe mechanism exposes a "data feed" which other contracts can regularly read for information. The data in this case is expected to change frequently, so client contracts must listen for updates to the data in the oracle's storage. An excellent example is an oracle that provides information on the latest ETH-USD price to users.

Request-response Oracles
Request-response oracles are an alternative oracle design pattern that allows smart contracts to request arbitrary data that is not provided by a publish-subscribe oracle. This design pattern is particularly useful when the dataset is too large to be stored on the blockchain or when users only need a small part of the data at any given time. In a request-response setup, the oracle has an on-chain component that receives data requests from client contracts and passes them to an off-chain node for processing. However, users initiating data queries must cover the cost of retrieving information from the off-chain source. Additionally, the client contract must provide funds to cover the gas costs incurred by the oracle contract in returning the response via the callback function specified in the request.

9.5 Security Attacks on Oracles

As the use of smart contracts and blockchain technology continues to grow, the importance of data oracles in facilitating communication between off-chain data sources and on-chain smart contracts cannot be overstated. However, data oracles are vulnerable to a range of potential attacks that can compromise the integrity and security of the system. These attacks include oracle spoofing, tampering, censorship, denial of service, front-running, bribery, extortion, and insider attacks. In order to ensure the reliability and accuracy of data oracles, it is crucial to implement strong security protocols and cryptographic techniques, as well as secure key management practices to protect against these types of attacks.

1. Oracle manipulation: This attack involves a malicious actor altering or manipulating the data provided by a data oracle, potentially leading to incorrect execution of a smart contract.
2. Oracle spoofing: This is a type of attack that involves a malicious actor presenting false or fraudulent data to a data oracle, potentially leading to incorrect execution of a smart contract. For example, an attacker could manipulate the data that is being fed into the smart contract, causing it to execute in a way that benefits the attacker. This could be done by presenting false data to the oracle, which would then be used to execute the smart contract. The attacker could then use this to their advantage, potentially causing financial harm to the victim.
3. Oracle censorship: This attack involves a malicious actor preventing a data oracle from accessing or providing certain data, potentially disrupting the execution of a smart contract.
4. Oracle Denial of Service (DoS): This attack involves overwhelming a data oracle with traffic or requests, making it unavailable to legitimate users and potentially disrupting the execution of a smart contract.
5. Oracle front-running: This attack involves a malicious actor manipulating the data provided by a data oracle to gain an unfair advantage in a smart contract execution.

6. Oracle bribery: This attack involves a malicious actor attempting to bribe or coerce a data oracle operator to provide false or fraudulent data, potentially leading to incorrect execution of a smart contract.
7. Oracle extortion: This attack involves a malicious actor threatening to disrupt or manipulate the data provided by a data oracle unless the operator pays a ransom or performs a specific action.
8. Oracle insider attack: This attack involves an insider (e.g., an employee or contractor) at a data oracle provider intentionally providing false or fraudulent data, potentially leading to incorrect execution of a smart contract.

It is important to protect against these types of attacks on data oracles, as they can have serious consequences, including financial losses, reputational damage, and the loss of trust in the system. This can be achieved through the use of secure cryptographic techniques, secure key management practices, and strong security protocols. We will discuss the countermeasures to prevent these attacks in the next section.

The following are some real-world examples of attacks on data oracles that have been reported:

Oracle Manipulation Attack: According to a report from Chainalysis, hackers stole $386.2 million from DeFi protocols in 2022 using "oracle manipulation" attacks. This type of attack involves artificially inflating the trading volume of a low-liquidity token on a DeFi protocol to spike its price. The hackers then use flash loans to secure the initial capital needed to inflate the token's trading volume, trade the designated token for a more stable crypto asset after pumping up the price, and leave the DeFi protocol insolvent. Chainalysis outlines the case of Avraham Eisenberg, who used $ten million worth of USDC to short 488 million MNGO (Mango governance token) and artificially inflate its price, leading to losses for Mango Markets.

(Devitt, 2023)

Other example includes Synthetix, a synthetic asset issuance platform built on Ethereum, which experienced an oracle manipulation attack which netted the attacker over 37 million sETH (Synthetic Ether) in June 2019. The true dollar value is difficult to calculate given the relative illiquidity of sETH on secondary markets. The attack was carried out by exploiting a vulnerability in the oracle used by Synthetix to manipulate the price of a synthetic asset and profit from the resulting arbitrage opportunity. The attack resulted in losses of approximately $one million (Todd, 2019).

Oracle spoofing attack: In 2020, an attacker exploited a vulnerability in the oracle used by the decentralized exchange (DEX) bZx to manipulate the price of a synthetic asset and profit from the resulting arbitrage opportunity. The attack resulted in losses of approximately $eight million. Although bZx was able to recover the funds, the reputation damage to the project is significant (Balakrishnan, 2020).

As oracles play more significant roles in the Web3 ecosystem, we expect more attacks and different and even new attacks on oracle. Web3 users and developers

need to be vigilant to get informed and Web3 project teams need to implement better oracle solutions to counter attacks.

9.6 Countermeasures to Oracle Security Attacks

As we discussed in the last sections, oracles are a potential weak point in the security of blockchain systems, as they can be susceptible to various types of attacks, such as spoofing, tampering, censorship, denial of service (DoS), front-running, bribery, extortion, and insider attacks. To address these security concerns, various countermeasures can be implemented.

Oracle manipulation: One potential countermeasure to prevent oracle manipulation is to use cryptographic techniques such as digital signatures and hash functions. Digital signatures can be used to authenticate the data provided by the oracle, ensuring that it has not been altered in transit. When the oracle signs the data, it produces a unique digital signature that can be verified by the smart contract to ensure that the data has not been tampered with.

Hash functions can be used to provide data integrity. A hash function takes an input and produces a fixed-length output, which is a unique representation of the input data. If the input data changes, even slightly, the resulting hash value will also change. By storing and verifying the hash value of the data provided by the oracle, smart contracts can ensure that the data has not been altered.

For example, consider a smart contract that executes based on the current temperature of a particular location. An oracle provides this data to the smart contract. To prevent oracle manipulation, the oracle can digitally sign the temperature data, and the smart contract can verify the digital signature. Additionally, the oracle can provide a hash value of the temperature data, and the smart contract can verify that the hash value matches the data provided by the oracle, ensuring data integrity.

Oracle spoofing: To prevent oracle spoofing, one potential countermeasure is to use multiple data sources and require consensus among them before allowing a smart contract to execute. By using multiple sources, the probability of all sources being spoofed simultaneously decreases. Consensus among the sources is then required to ensure that the data provided by the oracle is accurate and reliable.

There are several ways to implement this countermeasure. One way is to use a decentralized oracle network where multiple oracles provide data to the smart contract. The smart contract can then require a majority or consensus of the oracles to ensure the accuracy of the data. For example, Chainlink is a decentralized oracle network that provides decentralized data feeds to smart contracts and uses multiple oracles to ensure the accuracy of the data.

Another way to implement this countermeasure is to use multiple independent data sources outside of the blockchain network, such as API services or trusted third-party data providers. The smart contract can then compare the data provided by the different sources and require consensus among them before executing. For example, a smart contract that executes based on the current market price of a

particular asset can use multiple independent data sources, such as several financial data providers or exchanges, to ensure the accuracy of the price data.

Furthermore, blockchain networks can use different consensus algorithms to ensure the accuracy of the data provided by oracles. For example, Proof of Authority (PoA) consensus algorithm can be used to establish a set of trusted authorities, who then validate the data provided by the oracle. If a majority of the authorities agree on the data, the smart contract can execute.

Oracle censorship: To prevent oracle censorship, one potential countermeasure is to use decentralized oracle networks that are resistant to censorship, such as those based on peer-to-peer protocols. Decentralized oracle networks remove the need for centralized authorities to provide data to smart contracts, making them less susceptible to censorship. In a peer-to-peer network, nodes can communicate directly with each other, eliminating the need for intermediaries, and allowing nodes to act as both providers and consumers of data.

There are different ways to implement this countermeasure. One way is to use a blockchain network that includes a built-in decentralized oracle system. For example, Chainlink uses a decentralized oracle network that uses a combination of cryptographic proofs, reputation systems, and market incentives to ensure that the data provided to smart contracts is accurate and tamper-proof. In this system, data providers and consumers can interact directly without the need for intermediaries, making it resistant to censorship.

Another way to implement this countermeasure is to use a distributed oracle network that uses a consensus algorithm to validate the data provided by the oracles. In such a system, data providers can form consensus on the data they provide to smart contracts, eliminating the need for centralized authorities to validate the data. For example, Oraclize is a distributed oracle network that provides a trusted source of data to smart contracts by using a consensus mechanism that involves multiple oracles.

Additionally, blockchain networks can use peer-to-peer protocols, such as IPFS (InterPlanetary File System), to store and distribute the data used by oracles. IPFS allows data to be stored and accessed by nodes on the network, removing the need for centralized authorities to provide the data. By using peer-to-peer protocols, blockchain networks can ensure that the data used by oracles is resistant to censorship.

Oracle Denial of Service (DoS): To prevent Oracle DoS attacks, one potential countermeasure is to use distributed oracle networks that are resistant to DoS attacks, such as those that use redundant nodes or use sharding techniques to distribute the load. Distributed oracle networks have multiple nodes that can provide data to smart contracts, making them more resilient to DoS attacks. In a redundant system, if one node fails or is overloaded, the other nodes can take over and continue to provide data to smart contracts.

There are different ways to implement this countermeasure. One way is to use a blockchain network that includes a distributed oracle system with redundant nodes. In this kind of system, if one node fails or is attacked, the other nodes can continue to provide data, making it resistant to DoS attacks.

Another way to implement this countermeasure is to use sharding techniques to distribute the load. Sharding involves breaking down data into smaller, more manageable pieces and distributing them across multiple nodes. By doing so, the workload is distributed among multiple nodes, making it harder for an attacker to overwhelm any one node. For example, Augur, a decentralized prediction market platform, uses sharding to distribute the load across multiple oracles to ensure the accuracy of its predictions.

Additionally, blockchain networks can use a variety of DoS protection mechanisms, such as rate limiting, IP blocking, and anomaly detection, to detect and mitigate DoS attacks. These mechanisms can detect abnormal behavior and limit the amount of traffic a node can receive, preventing it from being overwhelmed.

Oracle front-running: One potential countermeasure to prevent oracle front-running is to use smart contract designs that are resistant to front-running. One such design is the use of randomness in smart contracts. By using randomness, it becomes difficult for an attacker to predict the outcome of a transaction and execute a similar transaction with a higher gas price. This makes it harder for the attacker to manipulate the outcome of the original transaction.

Another countermeasure is the use of commit-reveal schemes. In this approach, the transaction details are first committed to the blockchain without revealing them, and then the details are revealed later. This approach makes it difficult for an attacker to front-run the transaction because they cannot see the details of the transaction until it is too late to execute a similar transaction with a higher gas price.

Oracle bribery: One potential countermeasure to prevent oracle bribery is to use decentralized oracle networks that are resistant to bribery. These networks rely on multiple nodes to provide data to the smart contract, making it difficult for any single node to manipulate the data. Decentralized oracle networks can use different mechanisms to resist bribery, such as decentralized governance models or incentives to align the interests of oracle operators with those of the network.

Decentralized governance models involve the selection of trustworthy nodes based on a reputation system. The reputation of nodes is based on their past performance and the value of the data they provide. This ensures that only reliable nodes are selected to provide data to the smart contract. Additionally, nodes can be incentivized to provide accurate data by staking a certain amount of cryptocurrency as collateral. This incentivizes them to provide accurate data, as they risk losing their stake if they provide inaccurate data.

Another countermeasure is to align the interests of oracle operators with those of the network. This can be achieved by providing incentives to oracle operators to provide accurate data. For example, oracle operators can earn a portion of the fees generated by the smart contract. This incentivizes them to provide accurate data and ensures that they have a stake in the success of the smart contract. If an oracle operator provides inaccurate data, they risk losing their reputation score and their ability to earn fees in the future.

Oracle extortion: One important defense mechanism is monitoring of oracle nodes. This involves closely monitoring the activity and performance of each oracle node to detect any unusual behavior or signs of compromise. This can be done

through regular audits, security checks, and vulnerability scans. If any node is found to be compromised or at risk of compromise or extortion, it can be removed from the network to prevent it from providing inaccurate data to smart contracts.

Decentralized node operations can also be used as a defense mechanism against oracle extortion. In a decentralized network, multiple nodes are responsible for providing data to smart contracts, making it difficult for an attacker to gain control of the majority of nodes. This reduces the risk of a single point of failure and makes it more difficult for an attacker to manipulate the data provided to smart contracts.

Node governance is another defense mechanism that can be used to prevent oracle extortion. This involves implementing a governance model that ensures the oracle nodes are operated in a fair and transparent manner. This can include mechanisms for selecting trustworthy node operators, implementing staking or bonding mechanisms to incentivize accurate data provision, and regular audits and security checks to ensure that nodes are not compromised.

Oracle insider attack: To prevent such attacks, several defense mechanisms can be implemented. Here are some defense methods for preventing Oracle insider attacks:

Data Access Controls: One of the ways to prevent Oracle insider attacks is to implement data access controls that restrict the amount of data that an insider can access. By restricting the access to data, it reduces the potential for malicious actors to manipulate the data.

Regular Auditing and Security Checks: Regular auditing and security checks can be conducted to monitor the behavior of insiders and detect any signs of malicious activity. This can include checking for changes in access patterns or monitoring data requests to detect any unusual or unauthorized access.

Multi-party Computation: Multi-party computation (MPC) is a technique that allows multiple parties to compute a function or algorithm without revealing their inputs to each other. This can be used to prevent insider attacks by allowing multiple data providers to compute the final output of a smart contract without any one provider having access to all the data. This technique can make it difficult for an insider to manipulate the data without being detected.

Decentralized Oracle Networks: Decentralized oracle networks can help prevent insider attacks by relying on multiple data providers to supply data to the smart contract. This reduces the dependence on any single provider and makes it more difficult for insiders to manipulate the data.

Background Checks: Conducting background checks on employees or contractors who have access to sensitive data can help prevent insider attacks. By conducting a thorough background check, it is possible to identify any past criminal activity or questionable behavior that could indicate a potential insider threat.

In summary, the data oracle space has deficiencies and security risks and is open for new developments and players to enter. Decentralization is a core value for Web3 and blockchain applications and is the key for defense against oracle attacks. Innovative oracle and blockchain providers are expected to enter the space to meet market demand.

References

Arbol. (2021, January 12). Businesses and farmers can now hedge weather risk through the Arbol platform and Chainlink data. Arbol. Retrieved March 14, 2023, from https://arbolmarket.medium.com/businesses-and-farmers-can-now-hedge-weather-risk-through-the-arbol-platform-and-chainlink-data-d6f36506146c

Balakrishnan, A. (2020, September 14). bZx recovers $8.1M lost in third exploit. Crypto Briefing. Retrieved March 16, 2023, from https://cryptobriefing.com/bzxs-third-exploit-2020-ends-with-8-million-lost/

BusinessWire. (2017, June 15). AIG, IBM, standard chartered deliver first multinational insurance policy powered by Blockchain. Business Wire. Retrieved March 14, 2023, from https://www.businesswire.com/news/home/20170615005586/en/AIG-IBM-Standard-Chartered-Deliver-First-Multinational-Insurance-Policy-Powered-by-Blockchain

Cuenca, O. (2022, January 21). Etherisc launches automated blockchain travel insurance. ITIJ. Retrieved March 14, 2023, from https://www.itij.com/latest/news/etherisc-launches-automated-blockchain-travel-insurance

Devitt, C. (2023, March 10). Crypto hackers stole $386,200,000 from DeFi protocols via 'Oracle manipulation attacks' in 2022: Chainalysis. The daily Hodl. Retrieved March 29, 2023, from https://dailyhodl.com/2023/03/10/crypto-hackers-stole-386200000-from-defi-protocols-via-oracle-manipulation-attacks-in-2022-chainalysis/

Lohrmann, D. (2020, September 25). Could Estonia be the model for secure online voting? Government technology. Retrieved March 29, 2023, from https://www.govtech.com/blogs/lohrmann-on-cybersecurity/could-estonia-be-the-model-for-secure-online-voting.html

Sharma, S. (2022, September 8). *5 examples of Blockchain in supply chain management*. OyeLabs. Retrieved March 14, 2023, from https://oyelabs.com/examples-of-blockchain-in-supply-chain-management/

Shaw, J. (2020, January 27). Making smart contracts a reality with Blockchain technology. Oracle Blogs. Retrieved March 14, 2023, from https://blogs.oracle.com/blockchain/post/making-smart-contracts-a-reality-with-blockchain-technology

Todd, R. (2019, June 24). Synthetix suffers oracle attack, more than 37 million synthetic ether exposed. The Block. Retrieved March 29, 2023, from https://www.theblock.co/linked/28748/synthetix-suffers-oracle-attack-potentially-looting-37-million-synthetic-ether

Williams, A. D., & Kaplan, R. P. (2017, December 22). Diamonds on the Blockchain: Building a global digital ledger for valuable assets. DEEP Centre. Retrieved March 29, 2023, from https://deepcentre.com/wordpress/wp-content/uploads/2019/10/DEEP-Centre-Diamonds-on-the-Blockchain-December-2017.pdf