Blockchain Implementation for Secure Data Management with Genetic Storing Integration

Abdelrhman Elnamaki[1]
[1] Digital Career Institute, e23p07 , abdoelnamaki@gmail.com

August 17, 2024

Abstract

Blockchain technology provides a robust framework for secure data management, ensuring integrity, transparency, and immutability. This paper explores the implementation of a blockchain system in Python tailored for integrating an alternative chain to securely store genetic sequences extracted from plant disease reports, scientific papers, and datasets in each block. The system employs SHA-256 hashing for cryptographic security and DNA sequence storage to ensure long-term data preservation.

This project is a pivotal component of fulfilling graduation requirements from Digital Career Institute (DCI), demonstrating the practical application of blockchain technology and AI agents in the management and analysis of complex datasets. AI agents operating over the blockchain ecosystem enhance data management capabilities through automated analysis, validation, experimental contracts, and decision-making processes.

The applications of this integrated system span across healthcare, scientific research, and other data-intensive domains, showcasing blockchain's potential to improve data integrity, foster collaboration, and streamline information management. This study highlights how blockchain and AI technologies address contemporary challenges in data security and management, emphasizing their transformative impact on data-driven industries.

1 Introduction

Advancements in data-intensive fields such as Plants diseases and scientific research have underscored the critical need for secure and authentic data management systems. Traditional approaches often face challenges related to data integrity, transparency, and centralized control. Blockchain technology offers a promising solution by providing a decentralized and immutable ledger that ensures data integrity and security through cryptographic principles.

This project focuses on leveraging blockchain for secure data management, particularly in handling disease reports, scientific papers, datasets, and genetic

sequences . The integration of genetic data adds a unique dimension, requiring specialized methods for data storage and authentication. The system utilizes SHA-256 hashing for robust cryptographic security and incorporates DNA sequence storage to facilitate future advancements in genetic research and personalized medicine.

The blockchain implementation presented in this paper comprises two core components: the `Blockchain` and `GenChain`. The `Blockchain` facilitates continuous block generation, secure data validation using RSA cryptography, and management of diverse data types. Meanwhile, the `GenChain` specializes in storing and authenticating genetic sequences , ensuring data integrity through specialized hashing and DNA sequence conversion.

In this paper, we discuss the mathematical concepts and functions underlying RSA cryptography and SHA-256 hashing, providing a comprehensive understanding of their role in securing our blockchain operations and data storage. We also introduce the `Block` and `GenChainBlock` detailing their attributes and functionalities within the blockchain framework.

By demonstrating the implementation details and applications of blockchain technology in data management, this paper aims to showcase its potential in enhancing security, integrity, and efficiency across plant science field , and beyond.

2 Plant Chain

2.1 Blockchain

The `Blockchain` serves as the core framework for securely managing data operations within the system.such as plant diseases, scientific research, and beneficial datasets. The chain is meticulously designed to user contributions, enabling the reporting of plant diseases, submission of plant-related research, and sharing datasets crucial for biologists and plant scientists . This ensures a robust platform for collaborative knowledge exchange and comprehensive plant science advancements.

2.1.1 Functionality

- **Chain Operations**: The `Blockchain` facilitates seamless block management operations, including block creation, validation, and chain traversal. Each block in the chain is linked sequentially, with each block containing a cryptographic hash of the previous block, ensuring the integrity of the entire chain.

- **Data Storage**: It provides robust mechanisms for storing diverse data types such as disease reports, scientific papers, datasets, and translated genetic information. Each block within the chain can encapsulate multiple data entries, allowing for efficient and organized data management.

- **Cryptographic Validation**: Utilizing cryptographic methods such as SHA-256 hashing and RSA encryption, the `Blockchain` class ensures that each block's contents are securely hashed and validated. This cryptographic validation guarantees the authenticity and immutability of stored data, preventing tampering and unauthorized modifications.
- **Continuous Block Generation**: Through continuous block generation, the `Blockchain` class supports the dynamic addition of new data entries to the chain. This feature enables real-time data updates and enhances the chain's utility in applications requiring up-to-date information, such as disease surveillance and research collaborations.
- **B2DNA**: The system includes a unique feature serves as alternative chain called the GenChain, which tokenizes binary data into DNA sequences. This process, we call it B2DNA, facilitates the storage of potentially billions of data entries in a highly compact and efficient manner on the GenChain. By converting binary data into DNA sequences, the GenChain enhances data storage capabilities, ensuring that vast amounts of information can be preserved and accessed securely.

3 Components

3.1 RSA Encryption and Signing

3.1.1 Key Pair Generation

To generate an RSA key pair, the following steps are performed:

- **Generate large primes p and q**: Two distinct large prime numbers are generated.
- **Compute the modulus**:

$$n = p \times q \tag{1}$$

- **Compute the totient**:

$$\phi(n) = (p-1) \times (q-1) \tag{2}$$

- **Select the public exponent**:

$$e = 65537 \tag{3}$$

- **Compute the private exponent d**:

$$d \times e \equiv 1 \pmod{\phi(n)} \tag{4}$$

The public key consists of (e, n) and the private key consists of (d, n).

3.1.2 Private Key Serialization

The private key (d, n) is serialized to PEM format using PKCS8.

3.1.3 Public Key Serialization

The public key (e, n) is serialized to PEM format using the SubjectPublicKeyInfo format. This standard format ensures compatibility across different systems and applications.

3.1.4 Data Signing

To sign data using the RSA algorithm:

$$\begin{aligned} &\text{hash}(data) \rightarrow \text{Apply a hash function (e.g., SHA-256) to the data} \\ &\text{signature} = \text{hash}(data)^d \mod n \end{aligned}$$

PSS (Probabilistic Signature Scheme) padding is applied to ensure security against certain types of attacks:

$$\text{PSS padding} = \text{PSS}(\text{hash}(data)) \tag{5}$$

3.1.5 Signature Verification

To verify a signature:

$$\begin{aligned} \text{verification} &= \text{signature}^e \mod n \\ &= \text{hash}(data) \quad \text{if the signature is valid} \end{aligned}$$

PSS padding is used during verification to ensure the integrity of the hash function and the signed data:

$$\text{PSS padding} = \text{PSS}(\text{hash}(data)) \tag{6}$$

3.2 Block

The `Block` class is a set B, where each instance $b \in B$ is a tuple containing the attributes of the class:

$$b = (i, h_{\text{prev}}, t, d, b_{\text{type}}, k_{\text{public}}, h, c, \text{desc}, \text{info}) \tag{7}$$

where:

$$
\begin{aligned}
i &: \text{index (integer)} \in \mathbb{Z} \\
h_{\text{prev}} &: \text{previous\_hash (string)} \in \Sigma^* \\
t &: \text{timestamp (float)} \in \mathbb{R} \\
d &: \text{data (list of elements)} \in \mathcal{P}(D) \\
b_{\text{type}} &: \text{block\_type (string)} \in \Sigma^* \\
k_{\text{public}} &: \text{public\_keys (list of strings)} \in \mathcal{P}(\Sigma^*) \\
h &: \text{hash\_value (string)} \in \Sigma^* \\
c &: \text{creator (string)} \in \Sigma^* \\
\text{desc} &: \text{description (string)} \in \Sigma^* \\
\text{info} &: \text{additional\_info (dictionary)} \in \mathcal{F}(\Sigma^*, \mathcal{A})
\end{aligned}
$$

Thus, the set B of all possible blocks is defined as:

$$\text{B} = \{(i, h_{\text{prev}}, t, d, b_{\text{type}}, k_{\text{public}}, h, c, \text{desc}, \text{info}) \mid i \in \mathbb{Z},\ h_{\text{prev}} \in \Sigma^*,\ t \in \mathbb{R},\ d \in \mathcal{P}(D), b_{\text{type}} \in \Sigma^*,\ k_{\text{public}} \in \mathcal{P}(\Sigma^*),\ h \in \Sigma^*, c \in \Sigma^*,\ \text{desc} \in \Sigma^*,\ \text{info} \in \mathcal{F}(\Sigma^*, \mathcal{A})\ \} \tag{8}$$

Block Constructor

$$\text{init} : \mathbb{Z} \times \Sigma^* \times \mathbb{R} \times \mathcal{P}(D) \times \Sigma^* \times \mathcal{P}(\Sigma^*) \times \Sigma^* \times \Sigma^* \times \Sigma^* \times \mathcal{F}(\Sigma^*, \mathcal{A}) \to B \tag{9}$$

This function takes the inputs (corresponding to the parameters of the constructor) and produces an element $b \in B$.

Block Representation

$$\text{to\_dict} : B \to \mathcal{F}(\Sigma^*, \mathcal{A})$$

Given an instance $b = (i, h_{\text{prev}}, t, d, b_{\text{type}}, k_{\text{public}}, h, c, \text{desc}, \text{info})$, the dictionary representation is:

(10)

$$\text{to\_dict}(b) = \begin{Bmatrix} \text{"index"} & \mapsto i, \\ \text{"previous\_hash"} & \mapsto h_{\text{prev}}, \\ \text{"timestamp"} & \mapsto t, \\ \text{"data"} & \mapsto \{d_1, d_2, \ldots, d_n\}, \\ \text{"block\_type"} & \mapsto b_{\text{type}}, \\ \text{"public\_keys"} & \mapsto k_{\text{public}}, \\ \text{"hash\_value"} & \mapsto h, \\ \text{"creator"} & \mapsto c, \\ \text{"description"} & \mapsto \text{desc}, \\ \text{"additional\_info"} & \mapsto \text{info} \end{Bmatrix}$$

Serialization of Data

Given the block $b = (i, h_{\text{prev}}, t, d, b_{\text{type}}, k_{\text{public}}, h, c, \text{desc}, \text{info})$, the serialized data string S_d can be defined as:

$$S_d = \begin{cases} \text{to\_dict}(d_i), & \text{if } d_i \text{ has the method } \texttt{to\_dict} \\ \text{dict}(d_i), & \text{otherwise} \end{cases} \tag{11}$$

Where S_d is the serialization of each entry d_i in the data list d.

The full serialized block string S_b is then given by:

$$S_b = \text{str}(i) + h_{\text{prev}} + \text{str}(t) + \text{concat}(S_d) + b_{\text{type}} + \text{json}(k_{\text{public}}) + c + \text{desc} + \text{json}(\text{info}) \tag{12}$$

Where:

- $\text{str}(\cdot)$ converts the input to a string.
- $\text{concat}(S_d)$ concatenates all serialized entries S_d.
- $\text{json}(\cdot)$ converts the list or dictionary to a JSON string.

Hashing

The hash function H using SHA-256 is defined as:

$$H(x) = \text{SHA-256}(x) \tag{13}$$

Finally, the block hash h is computed as:

$$h = H(S_b) \tag{14}$$

Thus, the `calculate_hash` method is represented as:

$$\text{calculate\_hash}(b) = H\left(\text{str}(i) + h_{\text{prev}} + \text{str}(t) + \text{concat}(S_d) + b_{\text{type}} + \text{json}(k_{\text{public}}) + c + \text{desc} + \text{json}(\text{info})\right) \tag{15}$$

3.3 GENE CHAIN BLOCK

Represented as a set G, where each instance $g \in G$ is a tuple containing the attributes of the class:

$$g = (i, h_{\text{prev}}, t, s_{\text{dna}}, h) \tag{16}$$

where:

$$\begin{aligned} i &: \text{index (integer)} \in \mathbb{Z} \\ h_{\text{prev}} &: \text{previous\_hash (string)} \in \Sigma^* \\ t &: \text{timestamp (float)} \in \mathbb{R} \\ s_{\text{dna}} &: \text{dna\_sequence (string)} \in \Sigma^* \\ h &: \text{hash\_value (string)} \in \Sigma^* \end{aligned}$$

Thus, the set G of all possible genetic chain blocks is defined as:

$$G = \{(i, h_{\text{prev}}, t, s_{\text{dna}}, h) \mid i \in \mathbb{Z}, h_{\text{prev}} \in \Sigma^*, t \in \mathbb{R}, s_{\text{dna}} \in \Sigma^*, h \in \Sigma^*\} \tag{17}$$

Block Constructor

$$\text{init} : \mathbb{Z} \times \Sigma^* \times \mathbb{R} \times \Sigma^* \times \Sigma^* \to G \tag{18}$$

Representation

$$\text{to\_dict} : G \to \mathcal{F}(\Sigma^*, \mathcal{A}) \tag{19}$$

Given an instance $g = (i, h_{\text{prev}}, t, s_{\text{dna}}, h)$, the dictionary representation is:

$$\text{to\_dict}(g) = \begin{Bmatrix} \text{"index"} & \mapsto i, \\ \text{"previous\_hash"} & \mapsto h_{\text{prev}}, \\ \text{"timestamp"} & \mapsto t, \\ \text{"dna\_sequence"} & \mapsto s_{\text{dna}}, \\ \text{"hash"} & \mapsto h \end{Bmatrix}$$

4 Data Structures and Definitions

DataEntry

Our blockchain framework securely manages diverse data types critical for agricultural and scientific research. This includes detailed disease reports, scientific papers, and datasets, each structured to ensure integrity and accessibility.

Define the set of data entries E as:

$$E = \{\text{DiseaseReport}, \text{Dataset}, \text{SciencePaper}\}$$

where:

4.1 Disease Reports

d_r is characterized by:

$$\begin{aligned} d_r = (&\text{disease\_id}, \text{plant\_type}, \text{symptoms}, \text{diagnosis}, \text{treatment}, \\ &\text{date\_of\_incident}, \text{latitude}, \text{longitude}, \text{submitted\_by}, \\ &\text{notes}, \text{severity}, \text{environmental\_conditions}) \end{aligned}$$

These reports facilitate effective disease monitoring and response strategies.

4.2 Dataset

Dataset d_s is characterized by:

$$\begin{aligned} d_s = (&\text{dataset\_id}, \text{name}, \text{description}, \text{creation\_date}, \\ &\text{url}, \text{creator}, \text{data\_format}, \text{size\_bytes}, \\ &\text{license}, \text{tags}, \text{version}, \text{data\_sources}, \\ &\text{data\_quality\_metrics}, \text{hash\_value}) \end{aligned}$$

Datasets contain essential data for agricultural research, They support evidence-based research and analysis, ensuring data integrity and reproducibility.

Storing these data types on our blockchain enhances transparency, security, and collaboration in agricultural and scientific communities, fostering innovation and informed decision-making.

4.3 Science Paper

SciencePaper s_p is characterized by:

$$\begin{aligned} s_p = (&\text{paper\_id}, \text{title}, \text{authors}, \text{abstract}, \\ &\text{publication\_date}, \text{journal}, \text{url}, \text{keywords}, \\ &\text{citation\_count}, \text{related\_topics}, \text{doi}, \\ &\text{research\_field}, \text{methodology}) \end{aligned}$$

Scientific papers stored in the blockchain include research findings on plant diseases, This ensures transparent and traceable dissemination of scholarly information.

5 Main Chain

PlantChain $\mathcal{PC}$ is a list of blocks:

$$\mathcal{PC} = \{B_0, B_1, \ldots, B_n\} \tag{20}$$

where B_0 is the genesis block.

The genesis block B_0 is created as:

$$B_0 = \text{Block.create\_genesis\_block}(n, d) \tag{21}$$

where n is the name and d is the description.

Block Hash Calculation

The hash value h of a block B is computed using:

$$h = \text{SHA-256}(i \| h_{\text{prev}} \| t \| d \| \text{type} \| k \| c \| \text{desc} \| a) \tag{22}$$

where $\|$ denotes concatenation.

Mining and Genetic Chain Block Creation

To create a new genetic chain block G:

$$\begin{aligned} c_{\text{data}} &= \text{SHA-256}(\text{block\_data}) \\ b_{\text{data}} &= \text{binary\_data}(c_{\text{data}}) \\ s_{\text{dna}} &= \text{tokenize\_to\_dna}(b_{\text{data}}) \\ G &= \text{GenChainBlock}(i, h_{\text{prev}}, t, s_{\text{dna}}, \text{hash\_value}) \end{aligned}$$

where i is the index, h_{prev} is the previous hash, t is the timestamp, s_{dna} is the DNA sequence, and hash_value is computed similarly to block hash.

Adding Data Entries

Adding data entries to the blockchain involves:

$$\text{pending\_data} \leftarrow \text{DataEntry}(e) \text{ for } e \in E \tag{23}$$

and when mining:

$$B_{\text{new}} = \text{Block}(i, h_{\text{prev}}, t, d, \text{type}, k, h, c, \text{desc}, a) \tag{24}$$

where d includes data entries from pending_data.

Blockchain Validation

To validate the blockchain:

$$\text{For } i = 1 \text{ to } n : \begin{cases} \text{Check } h_i & = \text{SHA-256}(i \| h_{\text{prev}} \| t \| d \| \text{type} \| k \| c \| \text{desc} \| a) \\ \text{Check } h_{\text{prev}} & = h_{i-1} \end{cases} \tag{25}$$

where h_i is the hash of the block B_i and h_{prev} is the hash of the previous block.

6 GenChain

The `GenChain` is represented as a set C, where each instance $c \in C$ is a tuple containing the attributes of the class:

$$c = (n, d, v, \text{chain}, r, t) \tag{26}$$

where:

$$\begin{aligned} n &: \text{name (string)} \in \Sigma^* \\ d &: \text{description (string)} \in \Sigma^* \\ v &: \text{version (float)} \in \mathbb{R} \\ \text{chain} &: \text{list of GenChainBlock} \in \mathcal{P}(G) \\ r &: \text{running (boolean)} \in \{\text{True}, \text{False}\} \\ t &: \text{thread (threading.Thread) (optional thread)} \end{aligned}$$

Thus, the set C of all possible GenChain instances is defined as:

$$C = \{(n, d, v, \text{chain}, r, t) \mid n \in \Sigma^*, d \in \Sigma^*, v \in \mathbb{R}, \text{chain} \subseteq G, r \in \{\text{True}, \text{False}\}, t \text{ optional}\} \tag{27}$$

`Genesis_block`

The `create_genesis_block` method initializes the genesis block with fixed values. The genesis block b_g is defined as:

$$b_g = (0, h_0, t_0, s_{\text{dna}}, h_g)$$

(28)

Where:

$$\begin{aligned} 0 &: \text{index} \\ h_0 &: \text{previous\_hash} = \emptyset \\ t_0 &: \text{timestamp} \in \mathbb{R} \text{ (current time)} \\ s_{\text{dna}} &: \text{dna\_sequence} = \emptyset \\ h_g &: \text{hash\_value} = \text{computed} \end{aligned}$$

The method appends b_g to the chain.

`add_block`

The `add_block` method appends a block $b \in G$ to the chain. Mathematically, this can be represented as:

$$\text{chain}_{\text{new}} = \text{chain} \cup \{b\}$$

`mine_block` The block b is created as:

$$b = (i, h_{\text{prev}}, t, s_{\text{dna}}, h)$$

Where:

$$\begin{aligned} i &: \text{index} = \text{len(chain)} \\ h_{\text{prev}} &: \text{previous\_hash} \\ t &: \text{timestamp} \in \mathbb{R} \text{ (current time)} \\ s_{\text{dna}} &: \text{dna\_sequence} = \text{tokenize\_to\_dna(block\_data)} \\ h &: \text{hash\_value} = \text{hash\_block}(i, h_{\text{prev}}, t, s_{\text{dna}}) \end{aligned}$$

`hash_block`

The `hash_block` computes the hash value for a block:

$$h = H(i, h_{\text{prev}}, t, s_{\text{dna}})$$

Where:

$$H(x) = \text{SHA-256}(x)$$

and:

$$x = \text{str}(i) + h_{\text{prev}} + \text{str}(t) + s_{\text{dna}}$$

`tokenize_to_dna`

The `tokenize_to_dna` method converts a string to a DNA sequence:

$$s_{\text{dna}} = \text{convert\_binary\_to\_dna(binary\_data)}$$

(29)

Where:

$$\text{binary\_data} = \text{concat}\left(\text{format}(c, '08b') \text{ for } c \text{ in data}\right)$$

`list_blocks`

The `list_blocks` method provides a summary of the last 'max_blocks' blocks:

$$\text{block\_summaries} = \{\text{summary}(b) \mid b \in \text{chain}[start\_index :]\}$$

Where:

$$\text{summary}(b) = \begin{Bmatrix} \text{"index"} & \mapsto b.index, \\ \text{"timestamp"} & \mapsto \text{format\_timestamp}(b.timestamp), \\ \text{"dna\_sequence"} & \mapsto b.dna\_sequence, \\ \text{"previous\_hash"} & \mapsto b.previous\_hash, \\ \text{"hash"} & \mapsto b.hash \end{Bmatrix}$$

GenChain Integration: The `Blockchain` optionally integrates with `GenChain`, a component that tokenizes binary data into DNA sequences. This B2DNA (Binary to DNA) process allows for highly compact and efficient data storage. By converting binary data into DNA sequences, the `GenChain` can potentially store billions of data entries securely. The `GenChain` continuously mines new blocks and adds them to its chain, enhancing the overall data storage capacity and ensuring the preservation of vast amounts of information.

7 Applications

The `Blockchain` class finds diverse applications across industries, including:

- **Plant Disease Management**: Facilitating secure and traceable storage of plant disease reports, research data, and diagnostic results. Blockchain ensures data integrity, transparency, and collaboration among agricultural researchers, farmers, and biologists. This enhances the tracking and management of plant disease outbreaks and supports the development of effective treatments and preventive measures.
- **Scientific Research**: Enhancing transparency and reproducibility in plant science studies by securely storing and sharing research findings, datasets, and peer-reviewed publications. Blockchain's immutable ledger enables verifiable citations and intellectual property protection, fostering trust and collaboration within the scientific community.
- **Supply Chain in Agriculture**: Providing transparency and traceability in agricultural supply chain management by recording transactional data, product provenance, and compliance certificates. Blockchain mitigates fraud, counterfeit products, and enhances supply chain efficiency

and accountability, ensuring the quality and authenticity of agricultural products.

- **Genetic Data Storage**: Specialized applications include the storage and authentication of genetic data within the `GenChain` framework. The B2DNA (Binary to DNA) process tokenizes binary data into DNA sequences, allowing efficient and compact storage of potentially billions of data entries over the time . Blockchain ensures the integrity of genetic sequences, supports genetic storage collaborations.

8 Conclusion

This paper has presented a comprehensive exploration of blockchain technology's application in secure data management, with a specific focus on integrating genetic data storage. By leveraging Python and employing SHA-256 hashing for cryptographic security, our framework ensures data integrity, transparency, and immutability across diverse applications such as healthcare, scientific research, and data-intensive domains.

The implementation of RSA cryptography for data validation underscores our commitment to secure data handling practices, enhancing trust and reliability within the blockchain structure. Furthermore, the introduction of the B2DNA approach facilitates efficient storage of genetic sequences, showcasing our innovative approach to long-term data preservation.

Applications highlighted in this paper, including plant disease management, scientific research transparency, and genetic data storage, illustrate the versatility and profound impact of blockchain technology in safeguarding sensitive information and fostering collaboration across industries.

In conclusion, this study underscores the transformative potential of blockchain in revolutionizing data management practices, ensuring robust security measures and facilitating seamless data exchange in critical fields of study and industry applications.

9 References

1. Digital Career Institute. (n.d.). Digital Career Institute. Retrieved from `https://www.digitalcareerinstitute.org`

2. Mukherjee, A., Dutta, A., Bhaumik, C. (2019). Blockchain technology for secure data management: A survey. IEEE Communications Surveys Tutorials, 21(4), 3230-3243. `https://ieeexplore.ieee.org/document/9076545`

3. National Institute of Standards and Technology. (2015, August 5). Secure Hash Standard (SHS). Retrieved from `https://www.nist.gov/publications/secure-hash-standard`

4. Menezes, A., van Oorschot, P. C., Vanstone, S. A. (1996). Handbook of applied cryptography. CRC press.

5. Amin, M. R., Zhang, G., Sun, Z., Khan, F. H. (2021). Blockchain for Agri-Food Supply Chain Management: A Use Case Based Review of the Literature. Sustainability, 12(1), 40. https://www.mdpi.com/2077-0472/12/1/40

6. Zhang, Y., Wang, Z., Xu, X., Li, M. (2020). Blockchain Applications in Improving Scientific Data Management and Transparency. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8409015/

7. DNA storage: research landscape and future prospects. https://academic.oup.com/nsr/article/7/6/1092/5711038

8. Yan, Y., Yang, J., Li, J., Li, S., Liu, Z. (2022). A Survey on DNA-Based Information Storage. arXiv preprint arXiv:2205.05488. https://arxiv.org/abs/2205.05488

9. Li, Z., Liu, J., Sun, Y. (2017). Blockchain for provenance tracking: A survey from the perspectives of applications, techniques, and future research directions. Proceedings of the IEEE, 106(5), 977-1007. https://ieeexplore.ieee.org/iel7/6287639/9668973/09936616.pdf

10. Yusuf, A., Khan, F. I., Imran, M., Xia, L. (2021). Blockchain technology in agriculture: A systematic review of applications and challenges. Sustainable Agriculture Research, 10(5), 545-565. https://www.sciencedirect.com/science/article/pii/B9780128214701000033

11. Blockchain analytics and artificial intelligence. (2018). IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/abstract/document/8645631

12. Artificial Intelligence and Blockchain Integration in Business. (2022). Information Systems Frontiers, 1-18. Retrieved from https://link.springer.com/article/10.1007/s10796-022-10279-0