Lightweight Blockchain Prototype for Food Supply Chain Management

Alexey Rusakov*, Naghmeh Moradpoor (SMIEEE)*, and Aida Akbarzadeh‡
*School of Computing, Engineering and the Built Environment, Edinburgh Napier University, Edinburgh, UK
‡Faculty of Information Technology and Electrical Engineering,
Norwegian University of Science and Technology, Gjøvik, Norway

***Abstract*—The modern food supply chain often involves multiple layers of participants spread across different countries and continents. This complex system offers significant benefits to businesses worldwide; however, it also presents several challenges. One major problem is the inability to trace the product flow back to its origin, a critical issue in many industries. Another issue is the lack of trust among supply chain participants. Blockchain technology can help address these and other challenges faced by the supply chain industry. However, it is surprising that, globally, there are still not many examples of the technology's adoption, with most projects remaining in the pilot stage. This paper explores the field of custom blockchain design tailored to specific applications, with a focus on supply chain operations in the food industry. It includes the development of a lightweight yet fully featured Python prototype for a decentralized blockchain system. In addition to common features like block validation and state updates, the prototype includes a newly designed type of transaction tailored specifically for supply chain operations. These transactions eliminate the need for smart contracts, making the system more lightweight compared to general-purpose blockchain platforms such as Ethereum and less prone to security vulnerabilities. The prototype is designed as a public blockchain network, with Proof of Work selected as the consensus algorithm. The novelty of this research work lies in advancing the concept of a custom blockchain solution for the food industry. The key elements of the prototype have been unit tested. The overall evaluation was completed using a Python script that simulates product flow through an example supply chain, allowing product provenance to be determined by tracing the product flow back to its origin.**

***Index Terms*—Custom Blockchain, Provenance, Proof of Work, Product Flow, Blockchain from Scratch**

I. Introduction

The food supply chain often involves a complex network of participants spread across different countries and even continents. According to the authors in [1], these participants are primarily driven by profit-maximizing tendencies and competitive advantages, which can sometimes outweigh cooperation. However, the adoption of blockchain technology has the potential to transform this dynamic and bring multiple benefits to supply chain management. Although blockchain technology currently faces multiple challenges, such as scalability issues, data tampering before entering the blockchain, and regulatory compliance, it offers significant benefits for the supply chain industry such as shorter product traceability times, greater transparency, enhanced product provenance, and improved management of cold-chain food delivery.

According to the authors in [2], one of the major benefits blockchain technology offers to final consumers is the ability to determine and confirm the provenance of products, including the actual production date, time, and place of origin. Proving provenance is especially critical for certain types of products, such as medicine or food, as it allows consumers to verify whether the product is safe for consumption. The authors in [3] suggest that provenance certificates can be stored on the blockchain alongside the corresponding transactions.

Another major issue in the supply chain industry, aside from the provenance of products, is the time required to establish this provenance. Both authors in [4] and [5] highlight the potential for time savings in product traceability through blockchain solutions. Traditionally, it took approximately 7 days to trace the provenance, as each participant in the supply chain had to contact the previous participant until the product's origin was reached. In contrast, blockchain technology allows for determining the provenance of the same product within just 2 seconds.

Another benefit of adopting blockchain technology in the supply chain is improved transparency. According to the authors in [6], traditional supply chain systems often suffer from a lack of transparency, and blockchain solutions can help address this issue. Enhanced transparency allows stakeholders to view all participants in the supply chain and track the current status of shipments in real time. This improvement should also enhance accountability and facilitate quicker resolution of issues that arise during the product flow.

Blockchain adoption can also help address issues in applications requiring specific temperature conditions during product shipment or storage in industries such as food and healthcare. According to the authors in [7], a significant challenge in traditional supply chains is verifying whether the required temperature conditions were consistently maintained throughout all stages of the product's journey. A possible solution is to install temperature gauges that record temperature readings during the product's flow. The authors in [8] suggest that participants or consumers can request temperature readings, stored off-chain due to their size, and compare their hash values with those on the blockchain to verify data integrity.

In this paper, we developed a lightweight, fully-featured Python prototype for a decentralized blockchain system in the

food supply chain industry. Our proposed prototype includes a newly designed type of transaction specifically tailored for supply chain operations, eliminating the need for smart contracts. This approach makes the system more lightweight compared to general-purpose blockchain platforms like Ethereum and less prone to security vulnerabilities. There are four research questions we aim to answer through this study for the food supply chain industry:

- Which features from existing general-purpose blockchain solutions should be included, and which can be left out?
- Which consensus algorithm is most suitable for the proposed prototype?
- What API calls and transaction types are required for the supply chain application?
- What differentiates the proposed prototype from existing blockchain solutions?

The remainder of this paper is structured as follows: Section II reviews related work, Section III discusses the methodology and design, Section IV presents the implementation, Section V covers the results and evaluation, and Section VI concludes the paper with future work directions.

II. Literature Review

In this section, we review some of the related work on blockchain-based solutions for food supply chains, with a main focus on traceability.

In [9], the authors explore the problem of food supply chain traceability in the dairy sector and present their solution. The solution involves developing an Ethereum-based framework with three fully functional smart contracts that represent the interactions between participants in the supply chain. The system was tested and deployed on a local Ethereum blockchain network using Ganache-CLI. The authors demonstrate that the proposed smart contracts enable tracing product flow from raw materials to end customers. However, despite the flexibility and freedom smart contracts offer to program almost any logic, they can suffer from serious vulnerabilities, posing significant risks to businesses.

The authors in [10] explore a similar problem of product traceability in the dairy food supply chain but approach it from a different angle. Their main focus is on developing a system that consumes minimal computational power, making it more environmentally friendly. The proposed system is based on the Algorand Blockchain, which uses the Proof-of-Stake consensus algorithm. The system benefits from lower power consumption due to the selected consensus algorithm. It operates in real-time and successfully improves transparency among participants. However, their system requires the use of cryptocurrency which may create issues with exchange rates between the cryptocurrency and the local currencies used by participants. This could become problematic if exchange rates fluctuate, making transaction costs unpredictable.

The authors in [11] built an Ethereum-based prototype of a food traceability system that includes a consumer client and a user server. The system uses smart contracts instead of transaction records; however, the paper does not provide details about the algorithms used in the smart contracts. The authors highlight several advantages, including a higher degree of decentralization, increased security, and greater reliability. Another advantage is the potential reduction in transaction costs for small and medium-sized companies due to the system's features and flexibility. However, since the system is based on Ethereum and smart contracts, it may face potential vulnerabilities.

The paper [12] presents another example of building a food supply traceability system, which is based on a private blockchain framework, specifically Hyperledger Fabric. The authors combine IoT and blockchain technology to achieve the desired level of product traceability. The system includes an identification mechanism where each product has its own identifier, which can be verified by a smart contract. Their work addresses the problems of poor data sharing and lack of data security in traditional supply chain systems. However, since the system is based on a private blockchain framework, a limitation is that the system owner would need to identify all participants and invite them into the system, which could be a significant challenge in real-world applications.

The authors in [13] combine Radio Frequency Identification (RFID) and blockchain technologies in one system. The RFID reader interacts with the blockchain, writing information to the blockchain when a tag is read from the product. The tag can be attached to the paper packaging. The system is based on the Hyperledger Fabric platform, which verifies and manages the data and the reader. The main contribution of this paper is the development of a blockchain-based RFID tag. The authors emphasize the high level of security compared to traditional methods for tracing products. However, the private blockchain solution has limitations, including the need to identify all participants, which may not be feasible in a real-world environment. Another issue is the Byzantine Fault Tolerance (BFT) protocol used by the researchers, which has limitations and may not be well-suited for the application.

The authors in [14] propose an approach based on the Ethereum blockchain and smart contracts for tracking soybeans across the agricultural supply chain. The authors highlight that their system can be adapted for use in other agricultural supply chains. The system benefits from standard Ethereum trigger events, allowing participants to be notified when certain events occur. However, it may suffer from the same challenges as other Ethereum applications, such as vulnerabilities in smart contracts.

The conducted literature review shows that despite the numerous benefits blockchain technology can offer to the supply chain industry, real-life implementations remain very limited, with many still in the sandbox phase. One potential solution is to use a custom-designed blockchain tailored to a specific industry or application. Such a solution can benefit from a lightweight design by omitting unnecessary features, such as smart contracts. Removing these features not only makes the solution more efficient but also enhances

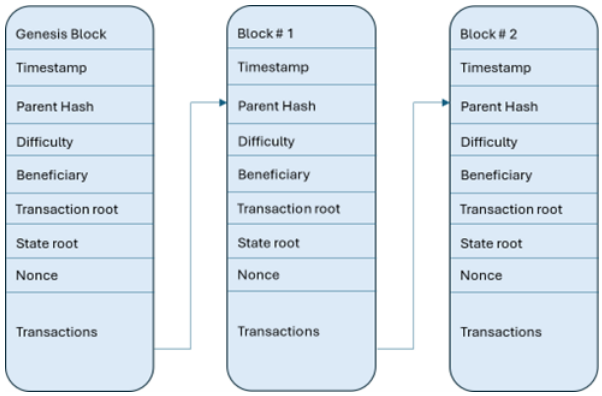

Fig. 1. Blockchain

security by reducing the risk of vulnerabilities. To address this crucial gap, in this paper, we developed a lightweight Python prototype for a decentralized blockchain system in the food supply chain industry. Our proposed system includes a newly designed type of transaction specifically tailored for supply chain operations, eliminating the need for smart contracts. Therefore, the novelty of this research lies in advancing the concept of a custom blockchain solution specifically for the food industry.

III. Methodology & Design

The section introduces the key concepts of the proposed blockchain solution, provides justifications for the technology selection, and references relevant literature where applicable.

A. Blockchain

A blockchain is a sequence of blocks linked by the hash values of the previous blocks (see Figure 1). The proposed prototype's block design includes a header and a body. The body holds the list of transactions, while the header contains the following fields:

- **Block number:** A sequential number that increments by one compared to the parent block's number.
- **Timestamp:** The timestamp recorded at the beginning of the mining process.
- **Parent hash:** The hash value of the parent block.
- **Difficulty:** An integer that controls the mining time for a given block.
- **Beneficiary:** The address (public key) of the miner who mined the block.
- **Transaction root:** The hash value of the transactions recorded in the block.
- **State root:** The hash value of the Trie data structure at the time of mining the block.
- **Nonce:** The solution to a cryptographic puzzle solved by miners.

B. Account

An account or wallet represents an entity or individual (user) who interacts with the blockchain [15]. Each account has a pair of private and public keys. The public key, either on its own or in a derived form, represents the address of the account and can be shared with others to send or receive transactions. In contrast, the private key remains secret and is used for signing transactions and/or data encryption. The account must also store a balance, which represents the amount of cryptocurrency available in the account.

C. Blockchain synchronisation

Synchronization is crucial as it represents the initial step for a new node to start interacting with the blockchain. Blockchain synchronization essentially means that a node, which has been disconnected from the network for a while, receives the most recent copy of the blockchain [16]. The synchronization process is also a part of new account creation to ensure that the new account has an up-to-date local copy of the blockchain.

D. Type of Blockchain

A private blockchain is attractive primarily due to its fewer security concerns; however, issues of trust and the difficulty in identifying all blockchain participants make this type of blockchain less preferable for the proposed prototype. A public blockchain has the advantage of being open to anyone and not owned by any single entity, making it more suitable for the proposed prototype. Although some general-purpose public blockchains suffer from scalability issues, the current prototype is still chosen to be a public blockchain. This is because scalability problems become critical only when the blockchain grows large and when the number of transactions processed per second becomes a significant concern. However, the number of participants in a global blockchain platform for supply chain operations is expected to be only a fraction of those on general-purpose platforms like Ethereum.

E. Consensus Algorithm

Among the three popular algorithms of Proof of Work (PoW), Proof of Stake (PoS), and Practical Byzantine Fault Tolerance (PBFT), PBFT is the only one designed for a private decentralized network where there is an owner and all other nodes are backup nodes. Since the proposed prototype is based on a public blockchain, PBFT cannot be considered. The remaining two algorithms, PoW and PoS, can both be implemented within public decentralized networks. However, because PoS is based on "stakes" and the proposed blockchain prototype is focused on supply chain operations, the concept of "stake" may not be well-suited to the prototype, at least in the initial phase of blockchain deployment. On the other hand, PoW is a proven solution employed in Bitcoin, making it a strong candidate for the consensus algorithm.

F. Block Validation

The block validation process is a key element of the entire blockchain operation, performed by all nodes before appending a new block to the blockchain [17]. Block validation is also executed as a routine task for all blocks during blockchain

synchronization. The block validation process should include some basic checks [18], such as verifying that the block number is correct, ensuring that the hash value written in the block matches the hash value of the parent block, and confirming that the PoW requirements are met for the given block. The authors in [19] also show that checking the block difficulty is necessary to avoid blockchain instability.

G. State

Each node in the blockchain network maintains a local copy of the entire blockchain, and therefore each node has all the transactions recorded within those blocks. However, efficiently retrieving this data can be a challenge. To address this, a tree-based data structure originally designed for efficient data queries and quick information retrieval is used. Bitcoin employs the Merkle tree as a data structure to capture the current state of the system [20], while Ethereum uses a modified version called the Merkle Patricia tree [21]. However, as pointed out by [22], the Merkle tree can be slow. Therefore, in the current prototype, a simplified version of the trie-based data structure will be used, as suggested by [23]. The trie-data structure is designed to store text data, allowing us to store transaction IDs and shipment details within it. Before being added to the tri, the text string is split into individual characters, each representing a node in the structure. Each node can have child nodes, creating a path that ends with a stored value. The tri must be updated with block data each time a new block is received to enable quick future retrieval.

H. Pub/Sub

The prototype must enable participants to receive and broadcast messages across the network. This is commonly referred to as a Pub/Sub service, where publishers broadcast messages and subscribers receive them. This feature is critical as it connects individual nodes and forms the blockchain network. Each node should be capable of sending and receiving messages. The mechanism should allow users to subscribe to specific channels and receive the corresponding messages. There should also be methods defined for handling received messages and for enabling nodes to broadcast messages across the network.The following channels must be defined within the blockchain network:

- **BLOCK:** Used by miners to send new blocks to peers. New blocks are transmitted in the body of the message in JSON format.
- **TRANSACTION:** Used to broadcast new transactions so that peers can add them to their local transaction pool.

I. Supply Chain Product Cycle

The product cycle for the proposed prototype needs to account for the typical processes in the supply chain industry, with a specific focus on the food sector. The term ”product flow” refers to the entire path of a product from its origin to the final consumer. In the food supply chain, the final consumer is typically represented by a grocery store or a restaurant. The product flow may involve multiple supply chain participants, which can be grouped into pairs. Each pair consists of a buyer and a vendor. The movement of the product from a vendor to a buyer is typically called a shipment.The proposed prototype should follow the workflow outlined below between a vendor and a buyer:

- The buyer initiates the shipment by sending a create shipment request to the vendor. Once this request is sent and validated, the amount of cryptocurrency is locked from the buyer’s account.
- The vendor then confirms the shipment through a confirm shipment request sent to the buyer.
- Once the shipment is received, the buyer sends a confirm delivery request to the vendor. After this request is validated by the system, the amount of cryptocurrency locked from the buyer’s account is transferred to the vendor’s account.

Each of these requests should be implemented via corresponding types of transactions, including a blocking currency transaction for locking cryptocurrency from the buyer’s account.

J. API Endpoints

API endpoints are URLs that allow users to communicate with the system. The system should include various API endpoints to enable users to perform standard blockchain operations, such as blockchain synchronization and creating new accounts, as well as supply chain-specific tasks like creating and tracking shipments. The names of the API endpoints should start with a forward slash to indicate that they are the endpoints of URLs.

a) Standard Blockchain Operations: The system should allow users to create accounts. Upon creation, users should receive both private and public keys. For the proposed prototype, a simplified method for creating a new account will be implemented by sending a GET request to the endpoint **/start**. This request will initialize the account creation process by sending the appropriate transaction type. For simplicity, this endpoint will also invoke methods that publish the transaction across the network and mine a new block. Additionally, as part of this process, all new accounts will receive an initial cryptocurrency balance of 100,000 units to facilitate blockchain operations. The API endpoint **/synchronize** should be developed to allow users to obtain the latest version of the blockchain after reconnecting to the network. The API endpoint **/mine** should be developed to enable users to participate in the block verification process and mine new blocks. This endpoint should also issue an appropriate reward transaction so that the miner who wins the competition receives a reward. Once a new block is mined, it must be added to the blockchain. The API endpoint **/account** should be available for users to view their current balance and address.

b) Supply Chain Operations: The proposed prototype should provide API endpoints that enable users to perform supply chain-specific operations, ensuring that the product

flow is accurately captured in the blockchain network. The API endpoint **create_shipment** should submit a transaction that captures the details of the shipment, including the addresses of the buyer and vendor, product description, quantity, price, and a reference to the previous shipment transaction. Additionally, there should be a specific transaction type that allows the system to debit the product's price from the buyer's account and block this amount until the delivery is confirmed. This blocking transaction should be sent immediately after the 'create shipment' transaction. Shipment creation should be allowed only for the buyer, not the vendor. The API endpoint **confirm_shipment** is the next step after the shipment is created. The vendor should confirm the shipment, so there should be a mechanism that permits only the vendor to confirm the shipment. A dedicated transaction type needs to be developed for this purpose. The API endpoint **confirm_delivery** is the final step in the shipment process. Only the buyer of the current shipment should have access and be able to confirm the delivery. A dedicated transaction type needs to be developed to confirm the delivery. Once the delivery is confirmed, the vendor receives the amount that was previously debited and blocked from the buyer's account. The API endpoint **provenance** should allow users to trace back and view the entire product flow down to its origin based on the entered shipment ID

c) Front-end Design: The front-end of the proposed prototype should allow users to perform all key operations related to the prototype, including submitting transactions and mining new blocks. The wireframe of the main HTML page is depicted in Figure 2. The top of the HTML page should display the current node ID and the address of the current user. The 'create shipment' section should allow users to enter data into text boxes for the parameters needed to create a new shipment transaction. The submit button should then send a POST request to the corresponding API endpoint. The result of the request should be displayed on the screen so that users can see whether the request was successful. The "confirm shipment" and "confirm delivery" sections should enable users to enter the shipment ID. The submit button should send the corresponding POST requests, and the results should be visible on the screen. The "Mine" button allows users to participate in the block validation process and broadcast new blocks across the network. All messages broadcast by peers should be displayed in real-time in a multi-line text box using WebSocket technology.

IV. Implementation

While Rust would be the most suitable programming language for our propsoed system, it generally requires more development time and is arguably not ideal for proof of concepts and prototypes. Therefore, Python was selected for this project. The prototype was developed using the following development environment, tools, and packages.

- Development Environment - WSL Ubuntu 24.04 LTS
- Programming Language - Python 3.12.3

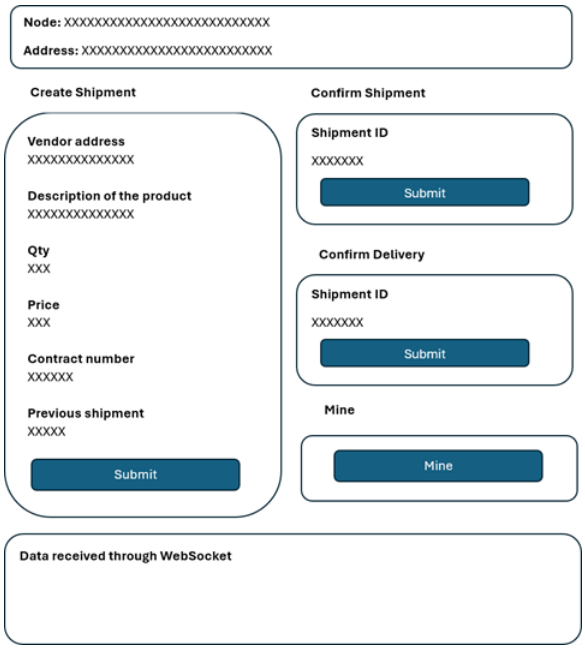

Fig. 2. Main webpage wireframe

- Web Framework - Django 4.2.13
- REST Framework - Django Ninja 1.1.0
- Cryptography package - cryptography 42.0.8
- WebSocket - Django Channels 4.1.0
- Pub/Sub - Redis 7.0.15
- Frontend - Bootstrap 5.3.3

The repositories such as [24], [25], and [26] were reviewed to explore potential approaches for implementing a blockchain solution. No code was copied from these repositories or any other sources. Due to the lack of space, we are not covering the general implementations common across different blockchain applications, such as accounts. The relevant code related to these general approaches can be found in our GitHub repository [27]. Therefore, this section presents the key Python methods designed to manage supply chain operations as follows.

A. Create Shipment Method

The **create_shipment method** is the key function that defines and creates a shipment transaction. The method first invokes another method to generate a new shipping transaction, and then calls a separate method to handle the currency blocking transaction Listing 1.

```
    txn_shipment = self.account.
        generate_new_shipment_transaction(
        vendor=vendor,
        buyer=buyer,
        product_description=product_description,
        qty=qty,
        price=price,
        contract_number=contract_number,
        previous_shipment=previous_shipment)

    # generate currency blocking txn
```

```
    txn_blck=self.account.
        generate_currency_blocking_transaction(
         amount=price, ref_txn_id=txn_shipment['body'
             ]['id'])

    return ([txn_shipment, txn_blck])
```

Listing 1. Create Shipment

The method that generates the new shipping transaction, Listing 2. It first sets the previous_shipment field to 'origin' if it is empty, indicating that this is the beginning of the product flow and the current shipment is the source of the product. It then populates the body of the transaction, Listing 6, signs the transaction, and returns a Python dictionary with two keys: 'body' and 'signature'.

```
def generate_new_shipment_transaction(self, vendor,
    buyer, product_description, qty, price,
                contract_number, previous_shipment):

    if previous_shipment == "":
        previous_shipment = 'origin'

    # [SKIPPED - Transaction body]

    signature = self.generate_signature(body)

    return {
        'body': body,
        'signature': signature
}
```

Listing 2. Generate Shipment Transaction

The body of the transaction, Listing 3, is represented by a Python dictionary with multiple keys. The key 'id' stores a pseudo-random unique ID. The type of transaction is 'CREATE SHIPMENT TRANSACTION', which helps differentiate this type of transaction from others. The dictionary also includes the addresses of the buyer and vendor, along with the key 'data', which can contain almost any data agreed upon between the buyer and vendor.

```
body = {
    'id': str(uuid4()),
    'type': TransactionType.
        CREATE_SHIPMENT_TRANSACTION.name,
    'vendor': vendor,
    'buyer': buyer,
    'previous_shipment': previous_shipment,
    'data': {
        'product_description': product_description,
        'qty': qty,
        'price': price,
        'contract_number': contract_number
    }
}
```

Listing 3. Shipment Transaction Body

B. Confirm Shipment Method

The confirm_shipment method is a wrapper function that generates the corresponding transaction, adds it to the transaction pool, and broadcasts the new transaction to peers through the Redis server, Listing 4.

```
def confirm_shipment(self, shipment_id):

    txn = self.account.
        generate_confirm_ship_or_deliv_txn(
        shipment_id=shipment_id,
        transaction_type=TxnType.
            CONFIRM_SHIPMENT_TRANSACTION)

    if not self.account.add_transaction_to_pool(txn)
        :
        return {
            'Error': 'Transaction is not valid',
            'Details': txn}
    self.redis.publish_transaction(str(txn))

    return (txn)
```

Listing 4. Confirm Shipment

The key part of the method that generates the 'confirm shipment' transaction, Listing 5. The transaction consists of a body containing the shipment ID, vendor address, and type of transaction. The method returns both the body and the signature of the transaction.

```
body = {
        'id': str(uuid4()),
        'from': self.address,
        'shipment_id': shipment_id,
        'type': transaction_type,
    }
    signature = self.generate_signature(body)

    return {
        'body': body,
        'signature': signature
}
```

Listing 5. Confirm Shipment Transaction

C. Confirm Delivery Method

The confirm_delivery method can only be invoked by the buyer and works similarly to the confirm_shipment method, with the only difference being that the type of transaction is set to 'CONFIRM DELIVERY', Listing 6.

```
def confirm_delivery(self, shipment_id):
    txn = self.account.
        generate_confirm_ship_or_deliv_txn(
        shipment_id=shipment_id,
        transaction_type=TxnType.
            CONFIRM_DELIVERY_TRANSACTION)

    # [SKIPPED]

    return (txn)
```

Listing 6. Confirm Delivery

D. API Endpoint /provenance

The API endpoint /provenance returns a Python list of shipments, starting from the shipment represented by the provided shipment ID and tracing back to the origin of the product flow.

The body of the API endpoint, Listing 8, primarily consists of a while-loop that retrieves the details of each shipment from the state and appends the current shipment to the list.

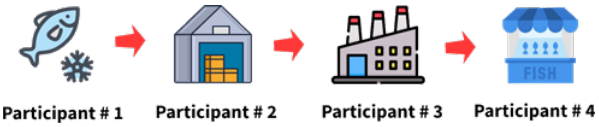

Fig. 3. Simulated Supply Chain

V. Results & Evaluation

We tested all aspects of the blockchain solution through four remote supply chain participants who interact to deliver a product from the source to its final destination. It demonstrates the ability to trace the product's origin based on the data available at its final destination. The participants in the simulated blockchain are illustrated in Figure 3. Participant #1 is a raw material supplier (frozen fish in this scenario), while Participant #4 is the final consumer, represented by the grocery store. Participants #2 and #3 are intermediate entities within the supply chain, which can be represented by, for instance, a food storage facility and a food processing center, respectively. Each participant has access to the blockchain via an API. However, for the purposes of the current simulation, all API calls will be simulated using a Python script. The simulation of the product flow begins with the preparation steps, where each participant is initialized. This is achieved by sending HTTP GET requests to the API endpoints /api/v1/start and /api/v1/account. These two API calls fully set up a new participant within the system and obtain their address for further reference, Listing 7.

```
res=requests.get(url_txt+"/api/v1/start",timeout=1)
res=requests.get(
    url_txt+"/api/v1/account",timeout=1)
res_dict=ast.literal_eval(res.text)
address=res_dict['address']
```

Listing 7. Initialization step

Once the buyers and vendors are set up, the simulation is ready to start. The product flow begins with Participant #2 (the buyer), who initiates the process by submitting the create shipment transaction, Listing 8. For the purpose of the current simulation, fields such as qty, price, and product_description will remain unchanged throughout the product flow. However, the field contract_number will vary, starting with Contract 1 for the first pair and Contract 2 and 3 for the subsequent pairs.

```
txn = {
    "vendor": address_vendor,
    "buyer": address_buyer,
    "product_description": "product_description_1",
    "qty": "999",
    "price": "500",
    "contract_number": "contract num 1",
    "previous_shipment": ""
}
```

Listing 8. Transaction - Create Shipment

The transaction is submitted via the HTTP POST method to the API endpoint /api/v1/create_shipment, Listing 9. Next, the string returned from the function call requests.post is converted into a Python dictionary, and the shipment_id is extracted. For simplicity, a new block is mined by the same node by sending an HTTP GET request to the API endpoint /api/v1/mine. Once this is completed, a new shipment is created in the state.

```
res=requests.post(
    url_buyer+"/api/v1/create_shipment",json=txn,
        timeout=1)
res_lst=ast.literal_eval(res.text)
shipment_id=res_lst[0]['body']['id']
print(f'\nShipment id: {shipment_id}')
res = requests.get(url_buyer+"/api/v1/mine", timeout
    =1)
```

Listing 9. Submitting the create shipment transaction

After the new shipment is created by Participant #2 (the buyer), Participant #1 (the vendor) will need to confirm the shipment. In the current simulation, this is done by sending an HTTP POST request to the API endpoint /api/v1/confirm _shipment, Listing 10. Once the shipment is confirmed by the vendor, a new block is mined by the same node for simplicity.

```
res=requests.post(url_vendor+"/api/v1/
    confirm_shipment",json=txn,timeout=1)
res=requests.get(url_vendor+"/api/v1/mine",timeout
    =1)
```

Listing 10. Confirming shipment

Finally, the delivery needs to be confirmed by Participant #2 (the buyer) by sending an HTTP POST request to the API endpoint /api/v1/confirm_delivery, Listing 11.

```
res=requests.post(url_buyer+"/api/v1/
    confirm_delivery", json=txn, timeout=1)
res=requests.get(url_buyer+"/api/v1/mine",timeout=1)
```

Listing 11. Confirming delivery

Once the shipment process between Participant #1 and #2 is completed, Participant #3 (the buyer) initiates a new shipment, and Participant #2, who now becomes a vendor, confirms it. Finally, Participant #3 confirms the delivery, completing the shipment. The same process—consisting of the three steps: create shipment, confirm shipment, and confirm delivery—occurs between Participant #4 (the buyer) and Participant #3 (the vendor). The product flow is considered complete once Participant #4 confirms the delivery. The process is straightforward and similar between each pair. What distinguishes each pair is that each uses the shipment_id from the previous pair as a reference to the prior shipment in each transaction. At the final step, the provenance of the product can be determined by sending an HTTP POST request to the API endpoint /api/v1/provenance and printing the results on the screen. The transaction ID is sent in JSON format, Listing 12.

```
res=requests.post(url_participant4+"/api/v1/
    provenance",json=txn, timeout=1)
r = ast.literal_eval(res.text)

for shipment in r:
    pprint(shipment)
```

Listing 12. Provenace Determination

VI. Conclusion & Future Direction

In this paper, a custom lightweight blockchain system tailored for supply chain operations has been built and tested. This prototype offers a blockchain solution with specially designed transaction types, specifically for the supply chain industry, focusing on the food sector. These new transactions enable users to track supply chain operations without requiring smart contracts, making the prototype lightweight and less vulnerable to security risks. In this paper, the following research questions were addressed through experiments:

- Which features from existing general-purpose blockchain solutions should be included, and which can be left out? Excluded from the design are smart contracts, which have been replaced with specially designed transaction types tailored for supply chain operations, making the prototype more lightweight. Another feature is the use of a Trie-data structure, which is simpler and faster than, for example, the Merkle Patricia Trie.
- Which consensus algorithm is most suitable for the proposed prototype? Based on the literature review, the PoW algorithm was chosen for the prototype as the most time-proven solution.
- What API calls and transaction types are required for the supply chain application? A set of API endpoints and corresponding transactions, such as creating shipments, confirming shipments, confirming deliveries, and tracking provenance, enables the complete cycle of product delivery between a vendor and a buyer. These API endpoints also allow for tracing the origin of the product.
- What differentiates the proposed prototype from existing blockchain solutions? The proposed prototype is more similar to Bitcoin in that it is tailored to a specific application, unlike Ethereum, which is a general-purpose blockchain platform. It also differs from Hyperledger Fabric, as it is based on public blockchain principles.

The proposed prototype can be improved in several ways. One area requiring special attention is data encryption. One possible solution is to use an additional layer, such as Zero-Knowledge Proof or a similar approach, which could be programmed and integrated into the prototype. Additionally, the prototype lacks an API endpoint to cancel an ongoing shipment and return the locked currency to the buyer.

References

[1] Sargent, C. S. and Breese, J. L. (2024). Blockchain barriers in supply chain: a literature review. Journal of Computer Information Systems, 64(1):124–135.

[2] Liu, B., Si, X., and Kang, H. (2022). A literature review of blockchain-based applications in supply chain. Sustainability, 14(22):15210.

[3] Azevedo, P., Gomes, J., and Romão, M. (2023). Supply chain traceability using blockchain. Operations Management Research, 16(3):1359–1381.

[4] Yiannas, F. (2018). A new era of food transparency powered by blockchain. In- novations: Technology, Governance, Globalization, 12(1-2):46–56.

[5] Galvez, J. F., Mejuto, J. C., and Simal-Gandara, J. (2018). Future challenges on the use of blockchain for food traceability analysis. TrAC Trends in Analytical Chemistry, 107:222–232.

[6] Tyma, B., Dhillon, R., Sivabalan, P., and Wieder, B. (2022). Understanding accountability in blockchain systems. Accounting, Auditing & Accountability Journal, 35(7):1625–1655.

[7] Chen, X., He, C., Chen, Y., and Xie, Z. (2023). Internet of things (iot)—blockchain-enabled pharmaceutical supply chain resilience in the post- pandemic era. Frontiers of Engineering Management, 10(1):82–95.

[8] Azzi, R., Chamoun, R. K., and Sokhn, M. (2019). The power of a blockchain-based supply chain. Computers & industrial engineering, 135:582–592.

[9] Casino, F., Kanakaris, V., Dasaklis, T. K., Moschuris, S., Stachtiaris, S., Pagoni, M., & Rachaniotis, N. P. (2020). Blockchain-based food supply chain traceability: a case study in the dairy sector. International Journal of Production Research, 59(19), 5758–5770.

[10] Varavallo G, Caragnano G, Bertone F, Vernetti-Prot L, Terzo O. Traceability Platform Based on Green Blockchain: An Application Case Study in Dairy Supply Chain. Sustainability. 2022; 14(6):3321.

[11] Q. Lin, H. Wang, X. Pei and J. Wang, Food Safety Traceability System Based on Blockchain and EPCIS, in IEEE Access, vol. 7, pp. 20698-20707, 2019.

[12] Zhang X, Li Y, Peng X, Zhao Z, Han J, Xu J. Information Traceability Model for the Grain and Oil Food Supply Chain Based on Trusted Identification and Trusted Blockchain. International Journal of Environmental Research and Public Health. 2022; 19(11):6594.

[13] Wang L, He Y, Wu Z. Design of a Blockchain-Enabled Traceability System Framework for Food Supply Chains. Foods. 2022; 11(5):744.

[14] K. Salah, N. Nizamuddin, R. Jayaraman and M. Omar, Blockchain-Based Soybean Traceability in Agricultural Supply Chain, in IEEE Access, vol. 7, pp. 73295-73305, 2019.

[15] Houy, S., Schmid, P., and Bartel, A. (2023). Security aspects of cryptocurrency wallets—a systematic literature review. ACM Computing Surveys, 56(1):1–31.

[16] Danzi, P., Kalor, A. E., Stefanovic, C., and Popovski, P. (2018). Analysis of the communication traffic for blockchain synchronization of iot devices. In 2018 IEEE international conference on communications (ICC), pages 1–7. IEEE.

[17] Lee, S. and Kim, J.-H. (2023). Opportunistic block validation for iot blockchain networks. IEEE Internet of Things Journal, 11(1):666–676.

[18] Antonopoulos, A. M. and Harding, D. A. (2023). Mastering bitcoin. " O'Reilly Media, Inc.".

[19] Sedlmeir, J., Zhou, Y., Papageorgiou, O., and Fridgen, G. (2024). The imminent (and avoidable) security risk of bitcoin halving. Available at SSRN 4801113.

[20] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf. [Online; accessed 16-July-2024].

[21] Wood, G. (2024). Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, pages 1–41.

[22] Knirsch, F., Unterweger, A., and Engel, D. (2019). Implementing a blockchain from scratch: why, how, and what we learned. EURASIP Journal on Information Security, 2019:1–14.

[23] Bodon, F. and Rónyai, L. (2003). Trie: an alternative data structure for data mining algorithms. Mathematical and Computer Modelling, 38(7-9):739–751.

[24] Git repo - Mastering Bitcoin Third Edition (2024). https://github.com/bitcoinbook/bitcoinbook/releases/tag/third_edition_print1. [Online; accessed 16-July-2024].

[25] Git repo - Build Ethereum From Scratch - Smart Contracts and More" (2024). Build ethereum from scratch - smart contracts and more. https://github.com/15Dkatz/build-ethereum-from-scratch. [Online; accessed 16-July-2024].

[26] Git repo - Commit-by-commit breakdown of Build a Blockchain & Cryptocurrency Full-Stack Edition (2024). Build a blockchain & cryptocurrency. https://github.com/15Dkatz/cryptochain_commits. [Online; accessed 16-July-2024].

[27] Git Hub repository of the codes https://github.com/alexey1095/bc-for-sc [Online; accessed 16-September-2024]