TECHNICAL UNIVERSITY OF DENMARK

Provenance tracking of physical objects using blockchain technology

May 19, 2022

Authors:	*Student No.:*
Jeff Gyldenbrand	s202790
Martin Sander	s202782
Rasmus Hjorth	s175120
Mathias Hansen	s175112

I. Abstract

Stolen belongings such as bicycles, jewelry and the like have long been a major problem for both the owner of the goods and insurance companies. Insurance companies pay out large sums of cash in compensation for stolen items. With the analog model of today, it can be difficult for individuals in private trading to determine if a given item is stolen or not, and if the person selling it is the legit owner. The paper propose an implementation of a decentralized blockchain solution to offer full transparency and track the provenance of items, which in the end will benefit both the people buying items, and the insurance companies.

***Keywords*—** Blockchain, Provenance, Ownership, Insurance, Crypto-currency,

II. Introduction and contributions

In this paper, we will propose a decentralized blockchain solution that can be used by the average citizen and shops that buy and sell products. It will be used to ensure that goods are not stolen and that they are being sold by a legitimate seller, and at the same time a solution that insurance companies can use to better track provenance, offer better terms for customers and generally be better protected against fraud.

To give the reader a better understanding of our solution, we will introduce the problem in section IV and then provide a short description of the state of the art, of existing solution that does related or similar things, in section V. With this foundation, we will first describe our core idea, the abstracted architecture and adopted design paradigms, in section VI and then get into our technical solution in section section VII. We have developed a beta-version of our solution and described a proof of concept with an explanation of our code, in section VIII. Ultimo we will end this paper with a conclusion and further work for the project, in section IX. Before diving into all of this, we will provide the reader with an introduction to blockchain and the usage of blockchain in general, in section III

Section	Mathias	Martin	Jeff	Rasmus
Abstract	25%	25%	25%	25%
Introduction	25%	25%	50%	0%
Blockchain in general	0%	100%	0%	0%
The problem	100%	0%	0%	0%
State of the art	0%	0%	25%	75%
Idea	0%	0%	100%	0%
Technical solution	0%	0%	0%	100%
Conclusions and further work	50%	50%	0%	0%
This list	100%	0%	0%	0%

III. Blockchain in general and its applications

Blockchain technology has been a hot topic ever since Bitcoin emerged in 2009, due to its wide span of usage possibilities. Blockchain is broadly defined as a ledger of information distributed across a peer-to-peer network providing an irreversible, protected and time-stamped record. In addition to transactions of cryptocurrency, the blockchain network has numerous potential applications in several industries. This includes use cases such as registering and clearing IP rights in IP-heavy industries, to provide evidence of first use and for proof of ownership of various physical items [10]. It is the ladder we will explore in this paper, where we will propose a solution to the problem of stolen bikes.

Allianz - uses blockchain to streamline cross-border auto insurance claims in Europe. There's a single source record of each claim. Processing time has been reduced to minutes, and costs have fallen 10%. So far it's being used by 25 Allianz subsidiaries to settle 850,000 claims.

Anthem - is testing the blockchain to try to speed up an arcane administrative process known as "coordination of benefits," which determines one's primary insurer. Through a shared ledger with Chicago-based Health Care Service Corporation for certain Medicaid members in Texas, the companies now make this determination in minutes or hours. Anthem's blockchain program processes around 3,000 to 5,000 verifications a month.

A.P. Møller-Mærsk - counts 250 ports and 20 ocean carriers using its proprietary TradeLens blockchain, which cuts time and reams of paperwork out of tracking containers as they move through global seaports. Sportswear giant Puma, which ships out of northern Germany, can now track a specific container in seconds rather than hours, according to Maersk. TradeLens, which Maersk co-developed with IBM in 2018, has tracked more than 55 million container shipments and is now being used by other shipping giants such as Germany's Hapag-Lloyd and Singapore's Ocean Network Express.

Boeing - Boeing is partnering with Canada's TrustFlight and developer RaceRocks to build a so-called digital aircraft record system that helps airlines keep up with required maintenance.

Coinbase – In addition to its trading business, its "Coinbase Cloud" software aims to help developers build crypto applications, and in October, it announced an NFT marketplace to compete with OpenSea. A month later, CEO Brian Armstrong told investors Coinbase NFT could become "as big or bigger" than its trading business.

Fujitsu - Runs a blockchain innovation lab in Brussels. In November, for example, water purification firm Botanical Water Technologies started building a trading platform using Fujitsu's in-house distributed ledger technology, which will allow sugar mills, distilleries and cola makers to sell or reuse the water they would normally discard during production. The platform, launching in April, will trace the water as it's purified, sold and delivered, and give companies the option to donate a portion of their purified water to water-scarce communities.

J.P. Morgan Chase - JPMorgan's Onyx Digital Assets network. By using smart contracts and JPM Coin, a digitized version of the U.S. dollar, Onyx repo trades settle in real-time instead of overnight, reducing settlement risk and manual processing. The intraday repo application has so far facilitated the movement of $230 billion in trades, completing about $1 billion in transactions a day. In June, Goldman Sachs began using Onyx [6].

IV. The Problem

Reports show 555,000 burglaries cases took place Denmark between 2011-2018, where only 29,000 of the police reports have led to a verdict or fine [20]. Often thieves are looking for electronics and jewelry but also products such as bicycles are coveted [23].

This is a huge social issue for citizens in Denmark and the statistics suggest that the police are having trouble helping the victims. But this is not only a social issue it's also an economic concern for insurance companies. A recent report suggest that the replacement value of the goods in Denmark was amounted to approx. 2 billion [25].

Since insurance companies' business plan is to sell insurance and hope that their replacement value isn't larger than the amount they sell insurance for. Insurance companies may therefore have an interest in lower their replacement value and this can happen based on the assumption that if no one is buying stolen items the thief wouldn't be stealing. Therefore a system where each second-hand buyer can validate items easily and conveniently may help reduce the interest of thieves to commit crimes. And by that therefore also affect the replacement value for insurance companies.

Since the market is immense, we have decided to narrow our focus down to bicycles. We will therefore develop a proof of concept which can verify the owner of each bike based on blockchain technology. We have decided to solve the problem with blockchain technology, because the ledger is immutable and spread across multiple computers. The owner of the bicycle would then always be able to validate ownership of the bicycle. We strongly believe that this will have an effect on how easy it can become to validate a product which is being sold from consumer to consumer.

V. State of the Art

At this point in time there is no existing solution that utilizes a decentralized system to keep track on the provenance history for psychical objects [13]. Our main goal is to create a decentralized system that is not depending on a single entity. With that said it has been possible to pin down a two companies that solves the same issue as this project.

The first company that we discovered was Wolli. Wollis vision is to create a platform where you can register belongings and validate second hand products beforehand [2]. Currently, Wolli is only able to handle bicycles but they are planning on expanding onto other products in the future. The main reason why our solution is different Wolli is that they are using Google Firebase [3] which means that their data foundations resolves around a centralized setup.

Another company that deals with the issue of being able to track provenance history and origin of objects are Everledger. Everledger offers three kind of products that is Capture, Amplify and Identify.
Capture is Everledgers own platform where you can find, purchase and transfer objects. Unfortunately the platform currently only caters to the Diamond Industry.
Amplify is a provenance API where you can search for object origin or product claims as well as transfer ownership of products.
Identify is a product offered to other businesses that helps companies to tell a more detailed provenance story by tracking their products using an intelligent labelling system (e.g. NFC tags).
Everledger achieves the same goal as we want to achieve with this project as their products is build on the fundamentals of blockchain technology to ensure a complete provenance history [14].
Based on our research it seems like Everledgers' products are State of the Art when it comes to using blockchain technology as the fundamental pillars in an application used track provenance and origin of objects.

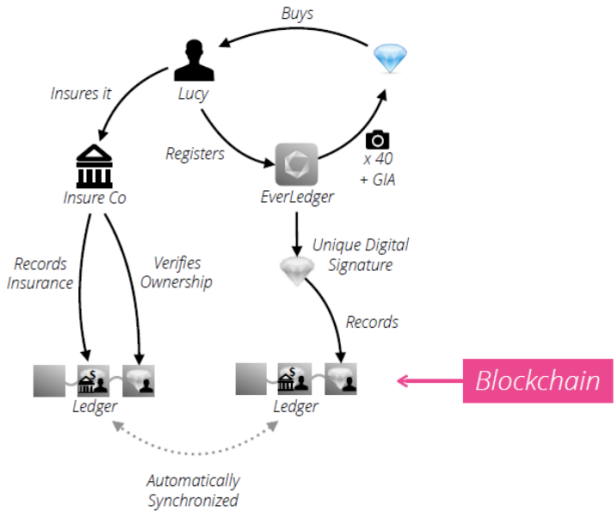

Fig. 1. Structure of the Everledger Platform [18]

One key difference between Everledger and the solution proposed in this project is that Everledgers' solution is based on a private permissioned blockchain approach. This is due to the fact that Everledgers' solution is powered by IMB Blockchain [18]. IBM Blockchain utilizes Hyperledger Fabric [22] which is build around private and permissioned network connections [19]. This makes Everledgers' platform a centralized system whereas this project strives to achieve the same main goal but with a decentralized structure.

A company that has implemented Everledgers' *Identity* product and technology in their business is the fashion label MYMCQ. All the products available from MYMCQ is marked with a unique digital identity in the form of a near-field communication tag. The digital identity is then used to track and trace the items journey from origin to its current owner [15]. MYMCQ addresses the issue of being able to track the provenance history of products but since they are using Everledgers' technology they are depending on a centralized system which differs their solution from the solution described in this project.

Other areas where distributed blockchain technology is used in relation to insurance is in the maritime industry, Guardtime [17] is a company that specialises in supply chains in the area of scale and performance, process automation, connectivity, provenance and authentication. Lemonade [24] for tenants and homeowners, Teambrella [29] for pet insurance, travel insurance and in healthcare [12].
From several of these we draw inspiration which inspired us to the main idea explained in the next section.

State of the Art in relation to the design and implementation of nodes in a blockchain network is a solution like Hyperledger FireFly which is a tool used to build and scale Web3 applications for enterprises.
Hyperledger FireFly 1.0 was introduced on the 13th of April 2022 and it is already used in production in a number of big blockchain consortia [28].

VI. The Idea

The idea of using blockchain technology in relation to insurance is not a new topic anymore, as mentioned in the previous section, blockchain is used for insurance in several industries. However, it is still in its early stages. All of the mentioned solutions are specific solutions tailored for specific businesses. We believe that there is a lack of a solution for the general citizen for the purchase of products and the assurance that a purchase of goods is legit and not stolen. Specifically, a solution that works in conjunction with the existing insurance companies, companies that don't rely on blockchain technology themselves. This can potentially reduce costs, improve risk assessment, and even enhance client on-boarding for the insurance companies as well as provide transparency and security for the consumer.

A. The key idea to our system

1) A regular buyer/seller & thiefs: The blockchain part of our solution is a fully transparent ledger. This ensures that a buyer is able to check the ownership of an item prior to purchasing it, regardless if it is a customer of an insurance company or not. This means that the buyer can check if an item is marked as stolen and check that the latest owner is the same person selling the item. With this solution we make it alot harder for thieves.

2) Bikeshops: are special in our blockchain, because they are the only one able to add new items (or bikes in this case) to the chain. A regular person is not allowed to add items to the system, only sell or buy existing ones.

3) Insurance companies: When an insurance company wants to be part of the system, they become nodes on the network. This means they are responsible for validating the chain. One could argue that this would make it a rather centralized system than a decentralized. However we believe that with enough different insurance companies on the network, being competitors of each other, would indeed make it decentralized.

4) Insurance customers: If you are a customer of an insurance company, you have the opportunity to become miner of the network and thereby earn points that can be used for, for example, cheaper insurances. To make this possible, we have created a currency called BikeCoin.

B. System architecture and adopted design paradigms

Our blockchain is build on the principle of an ordinary data structure; linked list, where each element in the list is called a block. From figure 2 we see a visualization of such a chain.

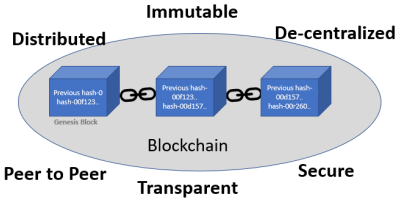

Fig. 2. Medium.com - Blockchain [21]

It is immutable, meaning that, once a block is added to the chain, it's not possible to change it. This is achieved with the principle of Stuart Haber and W. Scott Stornetta who propose a way of hashing and linking documents (or blocs) [27]. A simplified illustration of the idea is seen in Figure 3. As seen, the blocs of the chain have the fields; *Hash* and *Previous Hash*, when a block is added to the chain, a cryptographic hash of the data in the block is generated and added to that block, and the hash value of the previous block. In that way, if a malicious entity changes one block, the following blocs of the chain become invalid, thus he needs to change all the blocs. This is where the distributed part comes to play.

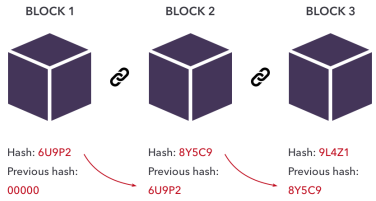

Fig. 3. Immutability secured by hashing and linking blocs

To avoid a single central point of failure (attack), and in general to create a decentralized system where no single authority has control, we need to distribute the blockchain amongst nodes on the network. As seen in Figure 4, (a) is completely centralized and would fail if the single authority was compromised. Vitalik claims the difference between (c) *"distributed means not all the processing of the transactions is done in the same place"* and (b) *"decentralized means that not one single entity has control over all the processing"* [5]. This is the key concept we have developed out system on. It becomes increasingly harder for an attacker, the more nodes that have a copy of the chain because these nodes are responsible for validating the correctness of the chain. This means that an attacker would need to outcompete at least 51% of the computing power of the network, in order to convince the rest of the nodes, that his chain is legit. This idea is taken from the original bitcoin system, created by anonymous Satoshi Nakamoto [26]. He/they propose the proof-of-work algorithm that we adapted to our system. Bitcoins PoW-algorithm works by:

> "1) New transactions are broadcast to all nodes.
> 2) Each node collects new transactions into a block.
> 3) Each node works on finding a difficult proof-of-work for its block.
> 4) When a node finds a proof-of-work, it broadcasts the block to all nodes.
> 5) Nodes accept the block only if all transactions in it are valid and not already spent.
> 6) Nodes express their acceptance of the block by working on creating." - Bitcoin whitepaper[26]

We draw great inspiration and has adopted a variant of this into our system.

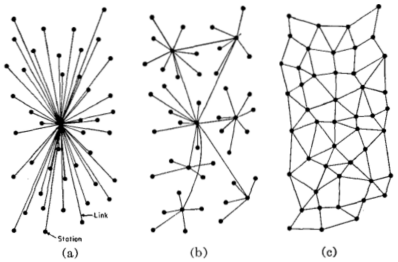

Fig. 4. (a) Centralized, (b) decentralized, (c) distributed [5]

C. Assumptions

We have made some assumptions in relation to the project, given its scope (5 ECTS), and short time. We have therefore not

implemented a frontend solution and therefore assume that a user of the system can access it with, for example, an app on their phone or access its functionality through a website where they log in. Likewise, we have not implemented any forms of authentication and therefore also assume that a possible frontend will require either login via a system, such as NEMID or that we implement a authentication solution ourselves. We also assume that our model is valid and insurance customers have a desire or financial incentive to be nodes on our blockchain and that the financial incentive holds, i.e. that it can pay off for customers to be nodes.

VII. Technical Solution

The implementation of a technical solution that addresses the issue and resolves the idea describes in section VI would mainly be based on existing technologies. It would be ideal for a framework such as Hyperledger FireFly that offers a variety of components needed to set up new nodes on a blockchain network. Hyperledger FireFly is a tech stack that contains pluggable components that can be used to build enterprise-ready Web3 applications. The main issue with using a framework such as FireFly is that it is build on top of existing public networks such as the Ethereum network which is not possible for this solution. The reason why it is not possible is due to the fact that there is a fee (gas price) associated with creating a new transaction on the Ethereum network and it has to be paid with Ether (Ethereum's own cryptocurrency). Unfortunately, Ethereum does not support the option of using different kinds of currencies to pay for the gas [1].

In the designed solution for this project, it is a necessity that it is possible to pay fees, gas and rewards with BikeCoin which then can be used to buy insurances from the insurance companies. The suggested solution will therefore be based on a new public blockchain network where it is possible to compensate the miners with BikeCoins.

Despite not being able to use Hyperledger FireFly directly, it is still a framework that can be used to seek inspiration on how to implement and design the components needed. Hyperledger FireFly is open-source and the source code is therefore publicly available as inspiration.

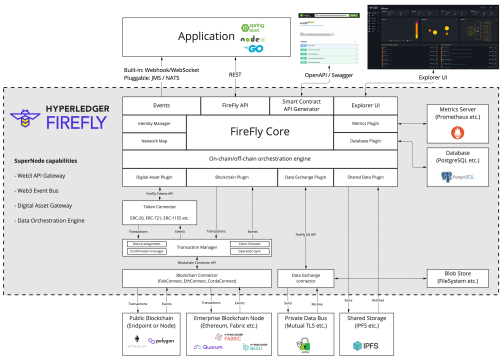

Fig. 5. FireFly architecture overview[11]

Hyperledger FireFly contains many useful components such as a Command-line Interface, a Transaction Manager, an Explorer UI, a Blockchain Connector and a HTTPS data exchange module.
The Command-line interface can be used to set up local development environments that can be used for building and testing.
The Transaction Manager is used to handle transaction related events such as submission of transactions and monitoring of blockchain operations.
The FireFly Explorer User Interface gives an easily accessible overview of everything that goes on inside the node as well as the system in general. It shows connected peers, blockchain structure, pending transactions, etc.
The Blockchain Connector is used to connect a node to a blockchain network. Since this solution will not use any of the existing networks the connector cannot be used directly in this project. Instead, it can be used to gain knowledge and inspiration on how Hyperledger connects to a blockchain.
The HTTPS data exchange module can be used to set up secure connections using private keys and certificates. A secure connection can be established both between nodes in the network as well as between nodes and clients.
Besides the previously mentioned components, FireFly can also help with generating OpenAPI/Swagger definitions for Smart Contracts as well as distributing them across the network so that every node knows how to contact the Smart Contract. Hyperledger FireFly contains even more components than the ones described above but these are some of the key components that could be quite useful in the implementation of this project.

Since blockchain is a technology that is built on other technologies and these technologies are necessary to implement the solution. We have decided to highlight and describe some of them in this section of the report.

1) Peer-to-peer: Before a user can validate his or her bicycle in the system. The Digital Ledgers have to be spread across the internet. This can be achieved based on the technology of peer-to-peer networks. This is a technology that makes each computer a server and client, this can therefore increase the security since the digital ledgers will be spread across multiple computers over the internet and thereby lower the risk of the digital ledgers being lost. But one of the concerns about peer to peer network is how to ensure that the other computer you are commincuation with is a real one

But one of the concerns about peer-to-peer networks is how to ensure that the other computers in the network you are communicating with is not an intruder. A solution can be to use mTLS which ensures that each computer is who they claim to be by verifying that they both have the correct private key. it happens based on their respective TLS certificates and provides additional verification.

mTLS is built on TLS which is an encryption protocol that uses public and private keys. public and private keys have two main functions, which is if something is encrypted with a public key, it can only be decrypted with a private key. This also counts the other way around so if something is encrypted with the private key it can only be decrypted with the public key. This means that if a server then decrypts a message that was encrypted with a public key, it then proves that it contains the private key This is exactly the technology that mTLS uses to ensure secure communication between multiple computers as a peer-to-peer network [7].

2) Consensus algorithm: Proof of Work: A consensus algorithm is used to achieve agreement on data distributed among systems. One of the key functionalities is that the algorithms are designed to achieve reliability in a network involving multiple unreliable nodes. To adapt to this, consensus algorithms believe that some systems will be unreachable and therefore some of the communications will be lost [9]. As a result of that, consensus algorithms must be fault-tolerant. They generally assume, for example, that only a portion of nodes will answer but demand a response from that portion, such as 51%, at a minimum.
Applications of consensus algorithms include:

- Deciding if a distributed transaction should be committed to a database.
- Delegates responsibility to different nodes for distributed task.
- Synchronizing state machine replicas and ensuring consistency among them.

Through the project, we had multiple consensus algorithms to choose from.

- Proof of Work
- Proof of Stake
- Proof-of-Burn

Proof of work is the most known type of consensus algorithm. In the system, miners receive rewards when they update the blockchain. They receive the rewards after they have used their computing power to solve a mathematical equation. Afterwards, the miners receive a reward such as Bitcoin or other cryptocurrencies for their job done. One of the disadvantages of PoW is that it cost a huge amount of power consumption and therefore isn't the most eco-friendly algorithm

Another consensus algorithm is Proof-of-Burn (PoB) which is an alternative algorithm to the Proof of Work (PoW) system. Proof of Burn tries to solve the problem with high energy consumption. It works by the principle of letting miners "burn" virtual currency. Miners are then able to write new blocks in proportion to the coins burnt. So the more coins the miners burn the greater the reward is to add a block. The miners are able to burn coins by sending the coins to a verifiable eater address, which is a randomly generated public address [16].

The last consensus algorithm we looked into was Proof of Stake which is an algorithm that gives miners rights proportional based on their stakes in the cryptocurrency. The blockchain is maintained by a group of randomly selected validators who is able to "stake" the native network tokens by locking them into the blockchain to produce and approve blocks. The problem is that this consensus algorithm makes the system more vulnerable to attacks based on how complex the construction is.

Based on the information and the research we decided to go with the Proof of work algorithm since it will be suited best for us because the miners only will get a benefit up to the amount that they have insurance for after that the miners would not benefit from it and by that it would require therefore require so much computer power to win the rewards of adding a block to the chain.

3) Cryptography:: Cryptocurrencies use cryptography to secure and anonymous transactions between the users of the system. This means that before a transaction can be made it's not necessary to receive information such as credit card company, bank, or any other third party in the middle. Through the project, we had multiple cryptography hashing algorithms to choose from.

- Message Digest Algorithm
- BLAKE
- Secure Hashing Algorithm

BLAKE has its advantages based on the speed it is hashing, but it was the least secure of these three [4]. We, therefore, decided to drop that. So did we with MD5 since it wasn't as secure as SHA-256. We, therefore, decided to go with SHA-256 for this proof of concept [8].

VIII. Proof of Concept

A Proof of Concept (PoC) application has been developed and implemented in order to validate the ideas behind this project as well as the designed solution.

A. Implementation

The Proof of Concept implementation consists of three mining nodes, a centralized wallet server and a client that can be used to initiate client side requests.

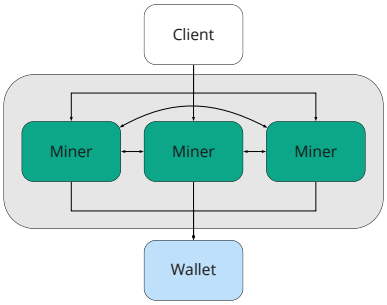

Fig. 6. PoC Application Structure

Each of the instances in Figure 6 are developed as stand-alone Web applications in Python using the web framework Flask. They grey area represent the boundaries of the P2P network that makes up the blockchain network. The nodes inside the grey area are connected and can communicate through http requests. The role of the client is to initiate new transaction requests, e.g. when a customer buys a bike from a bike shop a new transaction should be requested and added to the pool of pending transactions (mempool). After a miner receives the new transaction request it broadcasts the new transaction to all connected peers so they can add the transaction to their pool of pending transactions. The wallet instance is a simplified version of crypto-wallet and it is responsible of keeping track on the existing wallets as well as updating them according to the mined transactions. Each wallet contains an identifier, list of items attached to the wallet and a wallet type. The miner nodes are responsible of solving the cryptographic task associated with creating a new block. The first miner that solves the task is allowed to add a new block to the blockchain and broadcast its results to the network.
When a miner receives a broadcasted message regarding a mined block the miner will loop through the transactions inside the block and check if any of them matches a transaction from its mempool. If a transaction from the mined block matches a transaction in the pool, the transaction is removed from the mempool.

The PoC has been implemented with the main scenarios in mind and are as follows:

- S1: A bike shop can add new bikes to its wallet.
- S2: A person can buy a new bike from a bike shop.
- S3: A person can buy a bike from another person.
- S4: A person can validate that the seller is the real owner and that the bike is not stolen.
- S5: It should be possible to track and view the provenance history of a bike.
- S6: It should be possible to check the amount of BikeCoins added to a miners wallet.
- S7: BikeCoins can be used to buy an insurance (Not implemented).
- S8: All nodes should be able to validate their chains.

In order to test and validate that the PoC is able to handle the mentioned scenarios a Postman collection has been created. The collection consists of one folder for each of the scenarios that contains all the API requests that is involved in the specific scenario. In addition to this, the collection also contains a combined but simplified scenario [S9] that covers of all of the other scenarios in a single run. For all of the API calls there has been added Postman tests in order to validate that the requests returns the expected value. See appendix appendices A to H for test results.

Let us have a look at the simplified full scenario as an example of how the Proof of Concept operates. First of all it is only bike

shops who can add new bicycles to the chain. In this example the bike shop adds a total of 5 bikes. This is done by a bike shop owner requesting a transaction where new items are added to his wallet. A node registers the request and adds the transaction to its mempool as well as it broadcasts the received transaction to its connected nodes. In this example there is 3 running miner nodes.

```python
@app.route('/add_items_to_dealer', methods = ['POST'])
def add_items_to_dealer():
    json = request.get_json()
    items = json['items']
    wallet_id = json['wallet_id']

    response = requests.post(f'http://{wallet_url}/validate_dealer', headers=headers, json={"wallet_id": wallet_id})
    if(response.status_code != 200):
        return response.text, response.status_code

    response = 'Items has been added to wallet. Items have been added to the block with id: '
    transactions = []
    for item in items:
        t = {
            'sender' : "-",
            'receiver' : "Dealer",
            'data' : item,
            'timestamp': str(datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S:%f')),
            'status': "legit"
        }
        index = blockchain.add_pendingTransaction(t['sender'], t['receiver'], t['data'], t['timestamp'], t['status'])
        transactions.append(t)

    for node in blockchain.nodes:
        print(requests.post(f'http://{node}/broadcast_transactions', headers=headers, json=transactions).text)

    return jsonify(response + str(index)), 200
```

Fig. 7. Code snippet of adding new bikes to a bike shop

In order to actually get these new items added to the chain they need to be mined. The mining nodes tries to solve the cryptographic task of creating a new block containing the transactions in their mempool. The first node to succeed adds the block to the chain and broadcasts the mined block to its connected peers.

```python
@app.route('/mine_block', methods = ['GET'])
def mine_block():
    if(len(blockchain.pendingTransactions) == 0):
        return jsonify("No pending transactions"), 418

    response = requests.post(f'http://{wallet_url}/transfer_items', headers=headers, json={"pendingTransactions": blockchain.pendingTransactions})

    if(response.status_code != 200):
        return response.text, response.status_code

    previous_block = blockchain.get_previous_block()
    previous_proof = previous_block['proof']
    proof = blockchain.proof_of_work(previous_proof)
    previous_hash = blockchain.hash(previous_block)
    blockchain.add_pendingTransaction(sender = 'Chain Insurance Ltd.', receiver = miner_wallet_id, data = blockchain.miningReward,
                                      timestamp = str(datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S:%f')))
    block = blockchain.create_block(proof, previous_hash)

    is_valid = blockchain.is_chain_valid(blockchain.chain)
    if(not is_valid):
        return jsonify("The chain is invalid. Block is not added to chain"), 418

    response = {'message' : 'Tillykke, du har lige minet en block!',
                'index' : block['index'],
                'timestamp' : block['timestamp'],
                'proof' : block['proof'],
                'previous_hash' : block['previous_hash'],
                'transactions' : block['transactions']}

    for node in blockchain.nodes:
        requests.post(f'http://{node}/notify', headers=headers, json=block)

    return jsonify(response), 200
```

Fig. 8. Code snippet of mining a block

Next step in this scenario is that a customer buys a bicycle from the bike shop. The bike shop initializes a request for a new transaction containing the transfer of a bike from the shop's wallet to the customer's wallet.

```python
@app.route('/mine_block', methods = ['GET'])
def mine_block():
    if(len(blockchain.pendingTransactions) == 0):
        return jsonify("No pending transactions"), 418

    response = requests.post(f'http://{wallet_url}/transfer_items', headers=headers, json={"pendingTransactions": blockchain.pendingTransactions})

    if(response.status_code != 200):
        return response.text, response.status_code

    previous_block = blockchain.get_previous_block()
    previous_proof = previous_block['proof']
    proof = blockchain.proof_of_work(previous_proof)
    previous_hash = blockchain.hash(previous_block)
    blockchain.add_pendingTransaction(sender = 'Chain Insurance Ltd.', receiver = miner_wallet_id, data = blockchain.miningReward,
                                      timestamp = str(datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S:%f')))
    block = blockchain.create_block(proof, previous_hash)

    is_valid = blockchain.is_chain_valid(blockchain.chain)
    if(not is_valid):
        return jsonify("The chain is invalid. Block is not added to chain"), 418

    response = {'message' : 'Tillykke, du har lige minet en block!',
                'index' : block['index'],
                'timestamp' : block['timestamp'],
                'proof' : block['proof'],
                'previous_hash' : block['previous_hash'],
                'transactions' : block['transactions']}

    for node in blockchain.nodes:
        requests.post(f'http://{node}/notify', headers=headers, json=block)

    return jsonify(response), 200
```

Fig. 9. Code snippet of adding a new transaction to a node

In order for the transfer to take effect it needs to be mine according to Figure 8. After the first 3 steps the bike shop possess 4 bikes and a customer has 1. The blockchain in each node consists of 3 blocks. Next step in the example is for a customer to buy a bike from another customer. This is done by the owner of the bike requests to initiate a new transaction according to Figure 9 which is mined as described in relation to Figure 8.

The next part of the simplified scenario is that the newly acquired bike gets stolen and the owner reports this to the blockchain system by submitting a new transaction request to the system. This is done with the same endpoint as with Figure 9 but with the status *stolen*.

```python
@app.route('/verify_owner', methods = ['GET'])
def verify_owner():
    json = request.get_json()
    owner = json['owner']
    frame_number = json['frame_number']

    t_all = []
    for block in blockchain.chain:
        for transaction in block['transactions']:
            if(transaction['data'] == frame_number):
                t_all.append(transaction)

    res = sorted(t_all, key=lambda t: datetime.datetime.strptime(t['timestamp'], '%Y-%m-%d %H:%M:%S:%f'))
    if len(res) <= 0:
        return jsonify("No results found"), 200
    elif res[-1]['status'] == 'stolen':
        return jsonify("This bike is reported stolen")
    elif res[-1]['receiver'] == owner:
        return jsonify("Owner is verified!"), 200
    else:
        return jsonify("The suggested owner can not be verified as the real owner"), 200
```

Fig. 10. Code snippet of verifying bike information

When the thief then tries to sell the bike the buyer has the opportunity to validate that the seller is the correct owner of the bike and that the bike is not marked as stolen as shown in Figure 10. Besides being able to validate the ownership and status of a bike it is also possible to track the provenance history of a bike .

```python
@app.route('/provenance_history', methods = ['GET'])
def get_provenance_history():
    json = request.get_json()
    frame_number = json['frame_number']

    provenance_history = []

    for block in blockchain.chain:
        for transaction in block['transactions']:
            if(transaction['data'] == frame_number):
                provenance_history.append(transaction)

    sorted_history = sorted(provenance_history,
        key=lambda t: datetime.datetime.strptime(t['timestamp'], '%Y-%m-%d %H:%M:%S:%f'), reverse=True)
    return jsonify(sorted_history), 200
```

Fig. 11. Code snippet of retrieving provenance history

This is done by a user submitting a request to a node asking for it to return the provenance history of a specific bike. The customer retrieves a list showing all the events that bike bike has gone through such as trades, transfers and reports of theft which. can be used to track the origin as well as validate the timeline of the bikes existence.

B. Installation guide

The source code for the Proof of Concept implementation can be found here: GitHub Repository

In order to run the Proof of Concept application it is required that the user has installed Python with version 3.8 or later. It is also necessary that the user has installed the web application framework *Flask* with version 2 or later.

With the required installations in place it is now possible to run the program. This is done with 4 simple steps that includes running the following commands from the root of the project.

1) Start the centralized wallet server using:

```
python3 BikeCoin/wallet.py
```

The wallet server will then be accessible on localhost at port 5003.

2) Start up the three mining nodes using:

```
python3 BikeCoin/node.py <miner_id_1> <port_1>
python3 BikeCoin/node.py <miner_id_2> <port_2>
python3 BikeCoin/node.py <miner_id_3> <port_3>
```

In our example we have used the following input variables but these can be modified if needed.

Node	Miner ID	Port
1	Niels Christian The Miner	5002
2	Bubber The Miner	5001
3	Badekar The Miner	5000

If any of the values are changed the Postman test script will no longer be fully functional since it uses these values to perform requests and validate their responses.

3) Connect the nodes using:

```
./BikeCoin/connect_nodes.sh
```

Now that the nodes are connected the system is ready to use and therefore for the final step.

4) Run the Postman test collection: The system can be tested and validated using the Postman test collection which can be found in the *Test*-folder in the root of the GitHub project.

IX. Conclusion and Further Work

In this report, we presented the social and economic problems of handling the transferring of bikes from either shop-to-owner or owner-to-owner.
In the next section, we demonstrate the state-of-the-art landscape of technologies providing a provenance history for physical objects. The investigation showed that the companies Wolli and Everledger are attempting to solve the same issue, but currently no existing solutions exist providing provenance history with the use of an entirely decentralized system.
Finally, we present a decentralized blockchain solution that can be used by regular people and shops, to verify the legitimacy of ownership when transferring from shop to owner or owner-to-owner. This is done by displaying the idea and assumptions of the proof of concept to lay the groundwork for a presentation of the technical solution. The technical implementation showcases the application structure, as well as a detailed description of the main scenarios of the system including code snippets for each scenario.

Further work would be to implement smart contracts on the blockchain. This would allow insurance companies to customize insurance policies and other stuff which is not possible at the moment. A disadvantage of implementing this from the start is that we need to do a hard fork of the blockchain and insurance companies, customers, etc., need to use the new chain.

References

[1] Joshua (minimalsm). *Gas and Fees*. URL: https://ethereum.org/en/developers/docs/gas/.

[2] Wolli ApS. *Wolli ApS*. URL: https://wolli.dk.

[3] Wolli ApS. *Wolli ApS - FAQ*. URL: https://wolli.dk/resourcer/faq.

[4] blake2. *blake2*. URL: https://www.blake2.net/.

[5] Vitalik Buterin. *The Meaning of Decentralization*. URL: https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274.

[6] Michael del Castillo and Matt Schifrin. *Forbes Blockchain 50 2022*. URL: https://www.forbes.com/sites/michaeldelcastillo/2022/02/08/forbes-blockchain-50-2022/?sh=1cc8ba5a31c6.

[7] cloudflare. *cloudflare*. URL: https://www.cloudflare.com/en-gb/learning/access-management/what-is-mutual-tls/.

[8] codesigningstore. *Hash Algorithm Comparison*. URL: https://codesigningstore.com/hash-algorithm-comparison.

[9] TechTarget Contributor. *consensus algorithm*. URL: https://www.techtarget.com/whatis/definition/consensus-algorithm.

[10] Managing IP Correspondent. *Blockchain, IP and the fashion industry*. URL: https://www.managingip.com/article/b1kbpknf78y8tz/blockchain-ip-and-the-fashion-industry.

[11] Hyperledger FireFly Documentation. *Node Component Architecture*. URL: https://hyperledger.github.io/firefly/architecture/node_component_architecture.html.

[12] Epam. *Health-care*. URL: https://www.epam.com/how-we-do-it.

[13] Ethereum. *Ethereum: NFT - Physical items*. URL: https://ethereum.org/en/nft/#nft-physical-items.

[14] Everledger. *Everledger*. URL: https://everledger.io/technology/.

[15] Everledger. *Everledger and MCQ: bringing the future of fashion forward*. URL: https://everledger.io/everledger-and-mcq-bringing-the-future-of-fashion-forward/.

[16] JAKE FRANKENFIELD. *Proof of Burn*. URL: https://www.investopedia.com/terms/p/proof-burn-cryptocurrency.asp.

[17] GuardTime. *Whitepaper*. URL: m.guardtime.com/files/Blockchain%5C%20Designed%5C%20for%5C%20Supply%5C%20Chains%5C%20%5C%282%5C%29.pdf?fbclid=IwAR2oOvsUJ5afX33oxlJodTMp8P-z0FmeTsTstkXn3IzoiIMrZ-t8IcbH-vE.

[18] Carlo Gutierrez and Alex Khizhniak. *A Close Look at Everledger—How Blockchain Secures Luxury Goods*. URL: https://www.altoros.com/blog/a-close-look-at-everledger-how-blockchain-%20secures-luxury-goods.

[19] Hyperledger. *Hyperledger Fabric - Docs - Key Concepts*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/blockchain.html.

[20] Klaus Buster Jensen. *Indbrudstyve slipper godt fra 95 ud af 100 indbrud*. URL: https://www.dr.dk/nyheder/politik/folketingsvalg/nye-tal-indbrudstyve-slipper-godt-fra-95-ud-af-100-indbrud.

[21] Renu Khandelwal. *A Simple Guide to Understanding Blockchain*. URL: https://medium.com/swlh/a-simple-guide-to-understanding-blockchain-8dd09356b153.

[22] Matthew Kimmel. *IBM Blockchain*. URL: https://www.coindesk.com/company/ibm-blockchain/.

[23] PETER KRUIZE. "DET DANSKE HÆLERMARKED 2014 - 2019". In: (). URL: https://dkr.dk/media/7168/det-danske-haelermarked-2020-final.pdf.

[24] Lemonade. *Homeowner and renters insurance*. URL: https://www.lemonade.com/fr/en.

[25] Jonas Mannov. *Fakta om hæleri*. URL: https://dkr.dk/indbrud/fakta-om-haeleri.

[26] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. URL: https://bitcoin.org/bitcoin.pdf.

[27] W. Scott Stornetta Stuart Haber. *How to Time-Stamp a Digital Document - whitepaper*. URL: https://www.anf.es/pdf/Haber_Stornetta.pdf.

[28] Hyperledger FireFly Team. *Introducing Hyperledger FireFly 1.0: The SuperNode for Enterprise Web3 Applications*. URL: https://www.hyperledger.org/blog/2022/04/13/introducing-hyperledger-firefly-1-0-the-supernode-for-enterprise-web3-applications.

[29] Teambrella. *Whitepaper*. URL: https://blockchainlab.com/pdf/WhitePaper5.pdf.

APPENDIX

A. S1 - Test output

POST Add bicycles to dealer http://{{url}}:{{miner1}}/add_items_to_dealer / S1: Init dealer / Add bicycles to dealer 200 OK 79 ms 226 B

- Pass Response string contains block ID

GET Get Transactions http://{{url}}:{{miner2}}/get_transactions / S1: Init dealer / Get Transactions 200 OK 8 ms 704 B

- Pass Amount of transactions is 5 (matching number of bikes added to dealer)

GET Mine Block http://{{url}}:{{miner2}}/mine_block / S1: Init dealer / Mine Block 200 OK 41 ms 1.045 KB

- Pass Body contains congratulations message
- Pass Response contains 6 transactions
- Pass Body contains congratulations message
- Pass Response contains 6 transactions

GET Get Chain http://{{url}}:{{miner3}}/get_chain / S1: Init dealer / Get Chain 200 OK 7 ms 1.119 KB

- Pass No transaction in Genesis block
- Pass Mined block contains 5 bike transactions and 1 mining transaction

GET Get Wallets http://{{url}}:{{wallet_node}}/get_wallets / S1: Init dealer / Get Wallets 200 OK 7 ms 602 B

- Pass Dealer has 5 bikes

B. S2 - Test output

POST Add Transaction http://{{url}}:{{miner1}}/add_transaction / S2: Legit Buyer / Add Transaction 201 CREATED 43 ms 207 B

- Pass Response string contains block ID

GET Mine Block http://{{url}}:{{miner2}}/mine_block / S2: Legit Buyer / Mine Block 200 OK 140 ms 611 B

- Pass Body contains congratulations message
- Pass Response contains 2 transactions

GET Get Chain http://{{url}}:{{miner3}}/get_chain / S2: Legit Buyer / Get Chain 200 OK 3 ms 1.534 KB

- Pass No transaction in Genesis block
- Pass Chain contains 3 blocks
- Pass Mined block contains 1 bike transaction and 1 mining transaction

GET Get Wallets http://{{url}}:{{wallet_node}}/get_wallets / S2: Legit Buyer / Get Wallets 200 OK 3 ms 601 B

- Pass Buyer One owns bicycle1

C. S3 - Test output

POST Add Transaction http://{{url}}:{{miner1}}/add_transaction / S3: Legit Buyer to Buyer / Add Transaction 201 CREATED 20 ms 207 B

- Pass Response string contains block ID

GET Mine Block http://{{url}}:{{miner1}}/mine_block / S3: Legit Buyer to Buyer / Mine Block 200 OK 69 ms 615 B

- Pass Response contains 2 transactions
- Pass Response contains 2 transactions
- Pass Body contains congratulations message

GET Get Chain http://{{url}}:{{miner1}}/get_chain / S3: Legit Buyer to Buyer / Get Chain 200 OK 5 ms 1.952 KB

- Pass No transaction in Genesis block
- Pass Chain contains 4 blocks
- Pass Mined block contains 1 bike transaction and 1 mining transaction
- Pass Chain contains 4 blocks
- Pass No transaction in Genesis block
- Pass Mined block contains 1 bike transaction and 1 mining transaction

GET Get Wallets http://{{url}}:{{wallet_node}}/get_wallets / S3: Legit Buyer to Buyer / Get Wallets 200 OK 4 ms 601 B

- Pass Buyer Two owns bicycle1
- Pass Buyer Two owns bicycle1

D. S4 - Test output

GET Verify owner http://{{url}}:{{miner1}}/verify_owner / S4: Thief / Verify owner 200 OK 38 ms 217 B

- Pass Owner cannot be identified as the owner

POST Mark as stolen http://{{url}}:{{miner1}}/add_transaction / S4: Thief / Mark as stolen 201 CREATED 14 ms 207 B

- Pass Transaction is added to block

GET Mine Block http://{{url}}:{{miner3}}/mine_block / S4: Thief / Mine Block 200 OK 34 ms 623 B

- Pass Response contains 2 transactions
- Pass Body contains congratulations message

GET Verify owner http://{{url}}:{{miner1}}/verify_owner / S4: Thief / Verify owner 200 OK 4 ms 194 B

- Pass Bike reported stolen
- Pass Bike reported stolen

GET Get Chain http://{{url}}:{{miner1}}/get_chain / S4: Thief / Get Chain 200 OK 3 ms 2.378 KB

- Pass No transaction in Genesis block
- Pass Chain contains 5 blocks
- Pass Mined block contains 1 bike transaction and 1 mining transaction

E. S5 - Test output

GET Get provenance history http://{{url}}:{{miner1}}/provenance_history / S5: Provenance history / Get provenance history 200 OK 22 ms 624 B

- Pass Bicycle1's provenance history contains 4 events.

F. S6 - Test output

GET Check balance 1 http://{{url}}:{{miner1}}/wallet_balance / S6: Check miner wallet / Check balance 1 200 OK 18 ms 164 B

Pass Bubber The Miner has earned 200 BikeCoins while mining

GET Check balance 2 http://{{url}}:{{miner1}}/wallet_balance / S6: Check miner wallet / Check balance 2 200 OK 4 ms 164 B

Pass Niels Christian The Miner has earned 100 BikeCoins while mining

GET Check balance 3 http://{{url}}:{{miner1}}/wallet_balance / S6: Check miner wallet / Check balance 3 200 OK 3 ms 164 B

Pass Badekar The Miner has earned 100 BikeCoins while mining

G. S8 - Test output

GET Validate blockchain 1 http://{{url}}:{{miner1}}/is_valid / Validate blockchains / Validate blockchain 1 200 OK 7 ms 184 B

Pass Blockchain is valid

GET Validate blockchain 2 http://{{url}}:{{miner1}}/is_valid / Validate blockchains / Validate blockchain 2 200 OK 4 ms 184 B

Pass Blockchain is valid

GET Validate blockchain 3 http://{{url}}:{{miner1}}/is_valid / Validate blockchains / Validate blockchain 3 200 OK 3 ms 184 B

Pass Blockchain is valid

H. S9 - Test output

POST Add bicycles to dealer http://{{url}}:{{miner1}}/add_items_to_dealer / S8: Simplified full flow / Add bicycles to dealer 200 OK 47 ms 226 B

Pass Response string contains block ID

GET Mine Block http://{{url}}:{{miner1}}/mine_block / S8: Simplified full flow / Mine Block 200 OK 26 ms 1.046 KB

Pass Body contains congratulations message

Pass Response contains 6 transactions

POST Add Transaction http://{{url}}:{{miner1}}/add_transaction / S8: Simplified full flow / Add Transaction 201 CREATED 30 ms 207 B

Pass Response string contains block ID

Pass Response string contains block ID

GET Mine Block http://{{url}}:{{miner2}}/mine_block / S8: Simplified full flow / Mine Block 200 OK 129 ms 611 B

Pass Body contains congratulations message

Pass Response contains 2 transactions

POST Add Transaction http://{{url}}:{{miner1}}/add_transaction / S8: Simplified full flow / Add Transaction 201 CREATED 17 ms 207 B

Pass Response string contains block ID

GET Mine Block http://{{url}}:{{miner1}}/mine_block / S8: Simplified full flow / Mine Block 200 OK 74 ms 615 B

Pass Body contains congratulations message

Pass Response contains 2 transactions

POST Mark as stolen http://{{url}}:{{miner1}}/add_transaction / S8: Simplified full flow / Mark as stolen 201 CREATED 24 ms 207 B

Pass Transaction is added to block

GET Mine Block http://{{url}}:{{miner1}}/mine_block / S8: Simplified full flow / Mine Block 200 OK 35 ms 615 B

Pass Body contains congratulations message

Pass Response contains 2 transactions

GET Verify owner http://{{url}}:{{miner1}}/verify_owner / S8: Simplified full flow / Verify owner 200 OK 6 ms 194 B

Pass Bike reported stolen

GET Get provenance history http://{{url}}:{{miner1}}/provenance_history / S8: Simplified full flow / Get provenance history 200 OK 3 ms 624 B

Pass Bicycle1's provenance history contains 4 events.