Data sharing in the logistics domain

Christoph Braun, Björn Leuthe and Christian Merker

Karlsruhe Institute of Technology, Kaiserstraße 12, 76131 Karlsruhe

Abstract. Data sharing is the basis of a well working supply chain. Companies rely on information they receive from their business partners not only to operate efficiently in their daily business, but also when taking important business decisions. In addition, not only businesses but also society, clearly demanding more transparency regarding details on consumer products and their transportation, require a way of receiving and verifying information in a standardized manner. To solve this problem the present paper proposes the use and implementation of a Linked Pedigree, a form of a decentralized dataset representing the biography of a product. This Linked Pedigree is hashed, and its hashes then stored via a Smart Contract on a blockchain, to enable for a verification process of that biography data. The Linked Pedigree is structured according to the LinkedPedigree-Ontology developed by the authors. To evaluate the presented approach, different variations of Smart Contract implementations are compared. It's shown that a single contract instance for multiple products performs better than a contract instance for every single product. We also provide proof that the use of a Smart Contract in the first place is deemed necessary.

Keywords: Logistics, Linked Data, Smart Contract.

1 Introduction

1.1 Problem

Data sharing is the basis of well working supply chains. Companies must establish an information process based on what, how and when they plan to send to each other. Information and Know-How are valuable immaterial assets to a company and are therefore not likely to be shared with third parties. However, they must share information with suppliers and recipients in supply chains in order to merge their processes, which leads to the first problem. Sometimes this can even be critical information and supplier, or recipient can be a direct competitor to them. This means that parties of the supply chain must trust each other, which always imposes a risk. Of course, the parties still must trust their partners that they have the production capacity to enable a frictionless procedure during production. This kind of trust is not topic of the present paper but the trust in information mentioned before. It means, that the recipient of the information cares about the information as it was his own, including, that he does not use it in a bad way with the intention to harm his supply chain partner.

The second problem rises with the trend, that consumers increasingly want to know about the production place, the transportation stages, the time, the good has been stored at these stages and transportation time. Especially in the food and the pharmaceutical sector this trend is understandable and present. Consumers want to know, where the animal lived and how it has been treated in its life and how the meat or similar goods have come from the animal to them.

1.2 Objectives

Concerning the information process of supply chains, the first objective of the present paper is to enable information provision for the end-consumer in a 'trustless' way, so he can look up and verify every detail about the transportation process from the beginning to the end. The information stored as Linked Data by the parties of the supply chain is supposed to be unchangeable, so if one party tries to manipulate the information afterwards in order to hush up wrong storages of the goods or wrong treatment. This means that if some party has bad intentions and tries to change data and hush up defective goods, it can be remarked by the supply chain partners and the end-consumer.

An implementation of the presented model for demonstration purposes is also an objective of this work.

2 Motivation

2.1 Using Linked Data in the Logistics Domain

An increase in consumer conscience, about where their consumer goods come from and a demand on locally sourced products, makes crucial to provide customers with an easy and trustable method to look up where their product comes from. Additional with the possible creation of new regulations that would increases the demand for product documentation, making a solution for those demands.

Today's supply chain is very divers in its data structure, as most companies use an internal structure for more detailed information on a product. This data is not standardized among companies and isn't availed to the general public. Without a standard, it's impossible to make data easily available for all involving stake-holder in a supply chain. For this reason, we propose the use of RDF as a standard data model for all transactions. With this, a product can have a unique https URI that everyone could look up. This additional allows companies to exactly decide which information should be made public without granting access to a lager database. So, everyone only knows what everyone should know and can access what they need to know.

2.2 Proof and Trust in a trustless world

Most of the current work on Linked Data is still concerned with the lower layers (URIs, HTTP, RDF, …) of the Semantic Web Stack [1], thereby leading to elaborated standards for these layers ready for real world applications. However, when using Linked Data in real world (business) applications, the higher layers of the Stack, Proof and Trust, become a significant issue. While there are algorithms and tools to evaluate and even reason over Linked Data, verifying the authenticity and integrity of the received data poses a major challenge.

From a business perspective, it seems very risky to base any kind of business decision on Linked Data received from another party, since the data accessible via a URI can be changed arbitrarily anytime by its hosting party. There is no guarantee, that the data is and will remain true in a Tarskian sense, meaning that the data reflects reality truthfully.

For example, even a small business decision like accepting a package can cause numerous problems. Deciding to accept the package requires all its associated data at that time to be accepted as true. It would be irrational to make that decision based on information that is uncertain to be true in a Tarskian sense, because there would be no reasonable basis to justify that decision. Therefore, it would be necessary to trust in the data without being able to proof the data as true and without trust in the transmitting party. However, if data changes some time later maliciously or wasn't true by mistake in the first place, the decision might not be transparent or reasonable anymore.

Thus, there is a need for an authority, that can verify, whether or not the data is true in a Tarskian sense, or at least can proof, that the integrity of the received data is still intact. This authority basically acts as a notary providing the necessary proof for trusting in the received Linked Data. As a result, justified business decisions can be made

without trusting another party, except for the notarial authority. In the present paper, the notarial authority is decentralized and represented by a Smart Contract to reduce the need for trust in other parties even further.

3 Preliminaries

3.1 What is a Smart Contract?

In general, a Smart Contract is a computerized transaction protocol that executes automatically the terms of the contract when the required conditions are met.

It is therefore the purpose of Smart Contracts to satisfy common contractual conditions, e.g. payments, liens or information. With its automated contractual execution, a Smart Contract aims to minimize malicious and accidental exceptions as well as the need for trusted intermediaries. Associated economic objectives of Smart Contracts include lowering fraud loss, arbitration and enforcement costs, and other transaction costs. [2]

From a programming perspective, a Smart Contract is a piece of code, that is

a) stored on a blockchain (a distributed database with consensus on its status) and
b) executed in a decentralized manner (local execution, then synchronizing and consenting on the result, if any, with the network).

4 Proposed Approach

The proposed approach consists of three major parts, as depicted in **Fig. 1**: First, the interaction between members of the business network is explained in more detailed in 4.5. Second, the Linked Open Data Cloud relevant for the usage of Linked Pedigrees, which are introduced in 4.1 and expanded with its underlying ontology in 4.2. Lastly, the Smart Contract with its functionality, the hash verification process, will be laid out in 4.3 and 4.4.

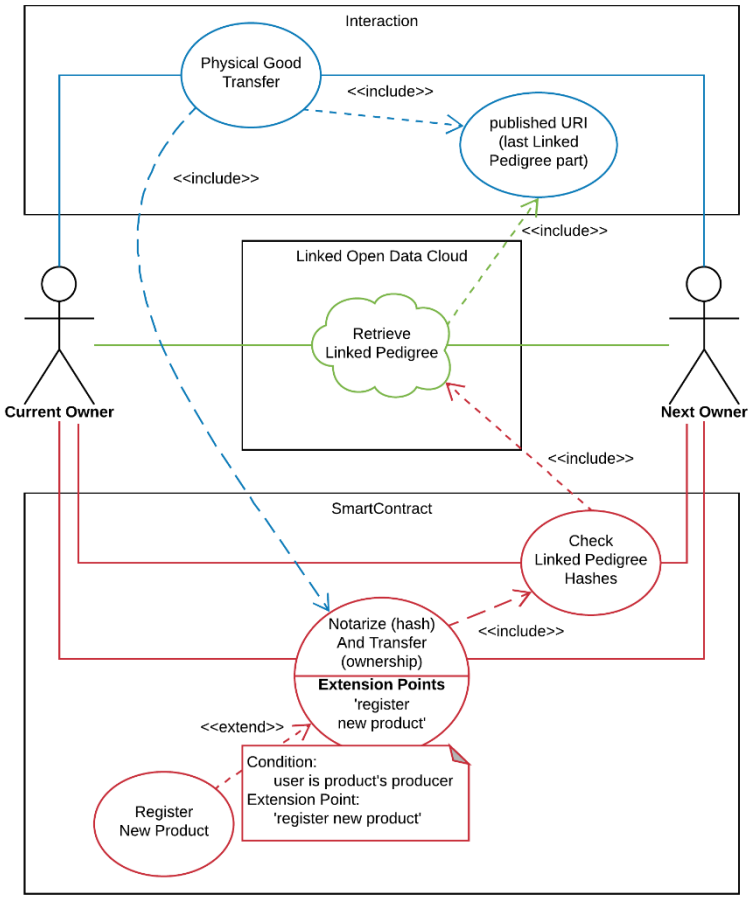

Fig. 1. Use case diagram

4.1 Linked Pedigree

As explained in 2.1, the use of Linked Data is beneficial for our problem. As a result, we need a data structure to support this idea and make the whole supply chain visible. In current supply chains the standard is to only receive information's one up and one down. To change this, we integrated a Linked Pedigree proposed by [3] into our Smart Contract structure. This event based decentralized databank of the products history, enables a user to trace back each Liked Pedigree to gain access to the complete product history. Solanki and Brewster also proposed OntoPedigree an ontology design patterns for those Pedigrees. [3] This ontology was not completely suitable for our pedigree as the most important part of the ontology the pedigree creation status was not completely compatible with our Smart Contract structure and some information where missing. For that reason, we adapted an own ontology presented in 4.2.

The Linked Pedigree uses the standard RDF data model with the ontology proposed in 4.2. Each dataset has a unique http URI. It also contains the URI of its predecessor and successor. With this we refer to other URI and follow the Linked Data principles.

A Linked Pedigree is built from the start of a supply chain, starting from the producer of the product and will end with the last vendor of the product. It will not only contain information about the previous and next owner but will also contain additional information about the product itself. With this information it should be possible to reproduce a complete biography of a product, if we trace back one Linked Pedigree. There are also two different kinds of the Linked Pedigree, the first being the birth certificate. The birth certificate is the first Linked Pedigree and represents the creation of the supply chain. Each following Linked Pedigree representing only one transfer from one part of the Supply chain to another is called transfer document.

To paint the idea more clearly, in **Fig. 2** graphical illustration of such a structure is presented.

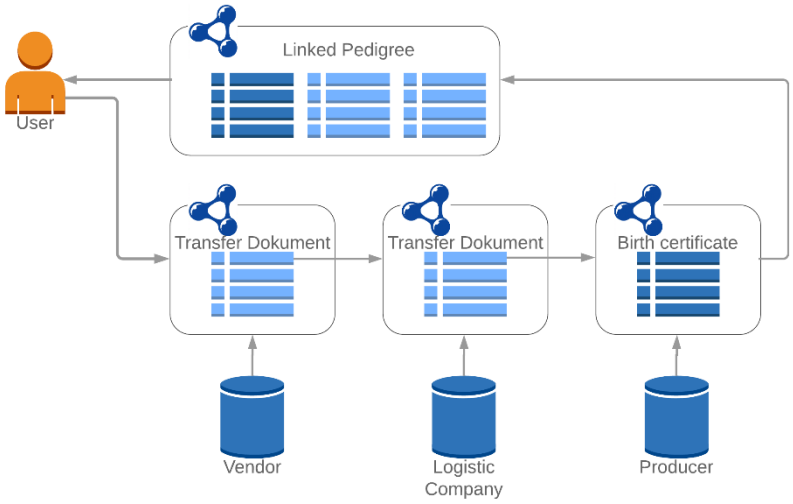

Fig. 2. Structure of a Linked Pedigree

4.2 Ontology

The data model that underlies and forms the base of the project is Linked Data. Ontologies enable describing the data in a way, that fits the claims defined at the beginning. Linked Data is a flexible data format that fits the use case at hand because of the distributed and adaptable information system.

In order to integrate the data, we are using, we propose OWL DL as it provides the constructions and classes necessary to model the data in this use case.

As an extension the developed ontology enhances the OWL inference as a logistic and supply chain specific data model described at the following page. The intention is to model the physical transportation process in a way, that allows to provide the data that is important to reach the target, which is transparency in terms of the data model.

The ontology has been designed in order to provide the information in the form of Linked Pedigrees. Therefore, not only the transfer point and time but also information that affects the supplier and the recipient are important to model. Furthermore, restrictions are suggested to avoid a wrong usage of the information process and at the same time create consistency between sometimes numerous members of the supply chain.

Transportation methods sometimes vary. This ontology can provide the data model referring to the exemplary supply chain introduced in the beginning. Extensions to the ontology can be created in order to customize the data model if e.g. in terms of transportation by flight or by train. The reason why this ontology is held general is because it has to fit into all forms of supply chains and especially to the use case at hand.

Ontology:

```
@prefix : <http://www.student.kit.edu/~ugyen/LDitld/ontologylpo#>.
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

:BirthCertificate a rdfs:Class.

:originProduct a rdfs:Class.

:Transferdocument a rdfs:Class.

:product a rdfs:Class.

:label a rdf:Property ;
        rdfs:domain :originProduct ;
        rdfs:range rdf:langString.

:created a rdf:Property ;
        rdfs:domain :originProduct ;
        rdfs:range :creation.

:creation a owl:Class;
        rdfs:subClassOf :transfer.

:createdBy a rdf:Property;
        rdfs:SubPropertyOf :transferredOwnershipTo;
        rdfs:domain :creation;
        rdfs:range :originFirm.

:time a rdf:Property;
        rdfs:domain :creation;
        rdfs:range xsd:date.

:place a rdf:Property;
        rdfs:domain :creation;
        rdfs:range :ThisPlace.

:ThisPlace a owl:Class.

:latitude a rdf:Property;
        rdfs:domain :ThisPlace;
        rdfs:range xsd:int.

:longitude a rdf:Property;
        rdfs:domain :ThisPlace;
        rdfs:range xsd:int.

:longitude a rdf:Property;
        rdfs:domain :ThisPlace;
        rdfs:elevation xsd:int.

:transfer a rdf:Property.

:ThisTransfer a rdfs:Class.

:transferredOwnershipFrom a rdf:Property;
        rdfs:domain :ThisTransfer;
        rdfs:range :Company.

:transferredOwnershipTo a rdf:Property;
        rdfs:domain :ThisTransfer;
        rdfs:range :Company.

:place a rdf:Property;
        rdfs:domain :ThisTransfer;
        rdfs:range :placeobject.

:placeobject a owl:Class.

:latitude a rdf:Property;
        rdfs:domain :placeobject;
        rdfs:range xsd:int.

:longitude a rdf:Property;
        rdfs:domain :placeobject;
        rdfs:range xsd:int.

:elevation a rdf:Property;
        rdfs:domain :placeobject;
        rdfs:range xsd:int.

:verifiedBy a rdf:Property;
        rdfs:domain :originproduct;
        rdfs:range xsd:string.

:previousPart a rdf:Property;
        rdfs:domain :Transferdocument;
        rdfs:range :Transferdocument.

:previousPart a owl:TransitiveProperty.

:previousPartHash a rdf:Property;
        rdfs:range xsd:string.

:time a rdf:Property;
        rdfs:domain :transfer;
        rdfs:range xsd:dateTime.

:transferPlace a rdf:Property;
        rdfs:domain :transfer;
        rdfs:range :ThisTransferPlace.

:ThisTransferPlace a rdfs:Class.
```

4.3 Hashing Linked Data

To ensure the validity of the Linked Pedigree, the RDF data is hashed. With this we will be able to compare the hash stored in the Smart Contract, with the hash of the current Linked Pedigree. If there is no match, we know that the data has been tampered.

RDF graphs have some properties that will create problems during hashing and must been addressed. One of them is the order of the triples. Even when the order of the triples changes the document should still result in the same hash. By simply string hashing the document, this cannot be guaranteed, as the order of the triples affects the resulting string and therefore its hash.

The second problem are blank nodes. Blank nodes are anonyms nodes without an IRI. So, without a global knowledge of the names of one of those blank nodes there is no possibility to recreate the hash again.

To solve both problems we will use the Babel software package for java that allows us to skolemize blank notes, to keep the isomorphisms even after labeling the blank notes and to create a canonical hash of the graph. [4]

4.4 Verifying Linked Data Integrity with a Smart Contract

Functionality. In the present paper, the Smart Contract represents a notarial authority providing proof for multiple Linked Pedigrees' Linked Data via the data's syntactic hashes.

In some way, the functional structure of the Smart Contract is similar to that of a distributed hash table. Hashes of Linked Data can be passed to the Smart Contract to be stored. Hashes cannot be altered or removed, once they are stored. After retrieving a hash from the Smart Contract, it can be compared to the hash directly generated from the Linked Data to verify that the data has not been changed since the hash had been passed to the Smart Contract.

To keep track of which hashes belong to which physical good's Linked Pedigree, the underlying physical good must be registered with the Smart Contract. Only after the hash of the good's birth certificate has been passed to the Smart Contract, any other hashes regarding that physical good can be stored.

When storing a hash, the Smart Contract assigns an index to keep the hashes in a sequential order. The index of the hash matches the position of the corresponding Linked Pedigree part in its Linked Pedigree "chain". This way, it is not only possible to compare the hashes, but also to check whether the hash is mapped to the correct Linked Pedigree part.

For example, the hash of a good's birth certificate is mapped to the index 0. The first transfer's hash is associated with the index 1 and so on. When checking the integrity of the Linked Pedigree, the hash for index 0 is retrieved from the Smart Contract, which can now be expected to be the same as calculated directly from the Linked Data of the

birth certificate. If the hashes don't match, the data received has been changed since the hash was stored via the Smart Contract.

Minimizing exceptions by design. Even if a changed data's hash would be passed to the Smart Contract, the index of that hash would not match the position of the corresponding Linked Pedigree part. It would not be retrieved when looking up the hash for the corresponding Linked Pedigree part, instead, the hash of the original data is retrieved. The indexed hashes provide therefore protection against hash doubling. Additionally, it is possible to identify which participant's data was changed because the index corresponds to the position of the changed Linked Pedigree part.

Retrieving hashes from the Smart Contract is unrestricted, so that anyone can verify the integrity of the corresponding Linked Data. However, storing new hashes via the Smart Contract is only allowed for the current owner of the physical good. The Smart Contract keeps track of the current owner of every physical good known to the Smart Contract. The current owner is only allowed to add one hash and is then forced to transfer the digital ownership of the good via the Smart Contract to the next owner. This security measure protects the Smart Contract from outside attackers trying to disable the Smart Contract by flooding it with hashes.

A dummy-owner-attack, where the current owner transfers the ownership to an entity controlled by the current owner to pass more than one hash to the smart contract, is already mitigated by the indexed hashes as discussed earlier.

By this design, malicious and accidental exceptions are minimized.

4.5 Application

Transfer Workflow. By integrating the previously introduced ideas into the use case at hand, a strict transfer protocol has been developed. When a transfer of a physical good is initialized, several actions need to be taken successfully, before the receiving party may accept the physical good.

First, the supplier generates the Linked Data associated with the transfer, which is the next Linked Pedigree part, and makes it accessible to the receiving party. The hash of that Linked Pedigree part is passed to the Smart Contract to be stored. Immediately, the digital ownership of the physical good is transferred to the receiving party via the Smart Contract.

Then, the recipient verifies that the information provided by the supplier is true in a Tarskian sense, meaning that the transfer is in fact carried out as stated in the generated Linked Pedigree part. Furthermore, the recipient traverses the single Linked Pedigree parts until the birth certificate of the physical good is reached. The Linked Pedigree's integrity is checked by comparing hashes directly calculated from each Linked Pedigree part with the hash for that part provided by the Smart Contract. Only if the integrity of the Linked Pedigree is intact and the generated transfer data is true, the physical good is accepted by the recipient and the transfer is completed. In case of inconsistencies during the transfer process, the physical good may be classified as defective. The

receiving party can therefore reject the physical good, return the digital ownership to the supplier and cancel the transfer.

Demonstration Tool. To showcase the approach proposed in the present paper, a minimal viable product (MVP) application has been developed. The implemented functionalities include retrieval of Linked Pedigrees, syntactic hashing of linked data and verifying the hashes of a Linked Pedigree against a Smart Contract on an Ethereum network. The Smart Contract can handle the hashes multiple Linked Pedigrees. Everything required to run the application is explained in detail on its repository. [5]

4.6 Applied Game Theory

Protecting published data. The transfer protocol has been designed to especially provide protection against tampering Linked Data, once the data has been published. There is no incentive for any party to change its data after publication, since the change is retraceable, and the corresponding party may be punished to any extent.

But even if a tampered Linked Pedigree would be propagated by a malicious coalition, the last participant in the supply chain has always the incentive to receive an intact good. It would be irrational for the last participant to join the malicious coalition because him/her would end up with a defect good, putting him/her in a disadvantageous position. Therefore, the last participant would reject the transfer. The participant before that also would then reject the previous transfer, since he/she does also not want to be the last one to receive the good, and so on. This leads to the conclusion that propagating a tampered Linked Pedigree would be irrational.

Therefore, there isn't any incentive to change data after publication or for propagating tampered data. With that, there is also no incentive to perform a dummy-owner- or double-hash-attack on the Smart Contract. Thus, the integrity of Linked Pedigrees is ensured.

Finding truth in a Tarskian sense? The system does not ensure truth in a Tarskian sense for the Linked Data published. Although it is not possible for a single malicious participant to publish false transfer data, since the recipient would reject the transfer, a malicious coalition between transfer participants might result in acceptance of false transfer data.

If the new current owner succeeds to perform the next transfer of the physical good according to the proposed protocol and in confirmation with real world contracts, the proposed approach will accept also the false transfer data as true.

The only case to identify false data in a Tarskian sense is, when a non-malicious party either somehow recognizes the data as false or real-world regulations or contracts intercept the transfer. That may be the case, if a delivery date cannot be met subsequently, or if an owner is not allowed to possess the good for more than a fixed time period.

There is no other internal possibility to verify truth in a Tarskian sense. That is a problem every system faces, that tries to reflect the real world digitally.

5 Evaluation

5.1 Smart Contract Variations

In the proposed approach, the Smart Contract is designed to be as flexible as possible regarding the way it can be deployed within a supply chain. The present paper recommends using a Smart Contract where multiple physical goods can be registered. One instance of that Smart Contract is deployed for verifying the integrity of Linked Pedigrees for all relevant physical goods in the business network. An alternative approach is using an instance of the Smart Contract for each physical good exclusively. In the following, the two Smart Contract variations introduced are compared.

Flexibility. Both alternatives of applying Smart Contracts are equally flexible regarding their functionality.

When using an instance of the Smart Contract for each physical good exclusively, the Smart Contract's functionality for an arbitrary good may be tailored to fit the participants needs. The same holds for using one Smart Contract instance as standard. If the standard functionality does not satisfy the participants needs, an upgraded version may be deployed.

It is in any case the obligation of the business partners to agree on which Smart Contract instance to use. Especially in large supply chains, this might cause significant overhead cost. Therefore, proposing a standard Smart Contract, that is already deployed and ready for usage, might facilitate business making in the network.

Storage Overhead. It is advantageous to not deploy a Smart Contract instance for every new physical good.

When using one Smart Contract for all relevant physical goods, only one single instance of the Smart Contract's code has to be stored in the database. Although the code seizes only to a few kilobytes, the storage overhead imposed by millions of code instances for Smart Contracts of every single good makes a significant difference.

In 2017 for example, DHL alone expected to deliver 8,5 million packages per day during Christmas season. [6] Scaling a few kilobytes by millions results in hundreds of gigabytes of Smart Contract code alone, generated per day. Code that is basically the same for every instance and therefore highly redundant.

The high storage overhead may discourage potential business partners to join the business network.

Operating Costs. It is economically preferable to use one Smart Contract instance for as many physical goods as possible.

Since Smart Contracts are stored in a distributed database and executed in a decentralized manner, the party deploying or calling the Smart Contract must pay for the computing power lend from other participants in the network. Therefore, the usage of Smart Contracts is in general associated with operating costs.

In the use case at hand, deploying a new instance of a Smart Contract to the distributed database is four times more expensive than registering a new good at an existing Smart Contract. If possible, reusing Smart Contracts is cost minimizing. **Table 1** provides an exemplary overview of gas costs estimated for Smart Contracts deployed to an Ethereum network.

Table 1. Estimated gas cost overview: Multiple registration Smart Contract and single good Smart Contract, estimated by remix.ethereum.org for the present implementation.

Action	Gas Estimates (multiple registration)	Gas Estimates (single good)
Creation/Deployment	379418	265953
Register new good	82221	N/A (create new contract)
Look up hash by ID	691	605
Store hash	41167	40921
Transfer ownership	20636	20506

If deployed to the public Ethereum network, costs would be transferable to regular fiat currencies. However, since cryptocurrencies like Ethereum are highly volatile, no reliable cost estimation is possible. Therefore, building a private Ethereum network proofs to be more controllable and predictable.

In a private Ethereum network gas costs still apply, but since the network is private, the cost of gas in monetary or computational terms can be set to a feasible level by the network. This way, operating costs can be minimized, and business remains profitable. However, there might be other network infrastructures, like a Hyperledger network, where Smart Contracts costs are even more controllable.

5.2 Omitting Smart Contracts

Without the existence of a Smart Contract, there's only the Linked Pedigree left. That eliminates the verification with a Smart Contract and as a result the trust in the system. Even if a Smart Contract-less system somehow saves hashes internally or guarantees correctness of data based on the trust people have in the company itself, there is no third party or network to check on a malicious coalition or individual. With that its possible for those parties to manipulate data without being caught. That eliminates the possibility for punishment. However, without punishment there is no negative outcome if someone manipulates the data. In case with manipulation the data a participant could gain something, there is a high incentive in doing so.

For the last party it is essential that the data is correct as he would be made responsible in the face of the public in case failures of the product become public and the real culprit can't be identified. For him it is not possible to check if the Linked Pedigree he receives is not tempered with or duplicated. Without the ability to find the culprit or check if the data is correct, there is no incentive for the last party to accept such a system compared to one with a verifiable Linked Pedigree.

With the incentive for manipulation and the unwillingness of the last party for such a system the Smart Contract based Linked Pedigree is superior in that regard.

In addition, such a system would also completely rely on the trust the public has in the companies governing the data. With more revelations coming out over the last few years over product tampering, be it cars or meat, this trust cannot be guaranteed in the future. As seen in 2.1 one of the reasons for such a system is the demand of the customers itself, so an acceptance from the customers is essential. Without giving the customer the ability to verify the data, this demand cannot be fulfilled. With that the system would not bear any additional value making it less desirable than a Smart Contract-based solution.

6 Conclusions and Future Work

Targets of the paper were to describe a way to make the transportation process of critical goods such as pharmaceuticals or food more transparent for the end-consumer. Subsequent manipulation of the information should be avoided in order to protect the consumer. This has been realized by using technologies like blockchain in combination with the data model Linked Data. It improved the information process of the transportation either for the end-consumer but also for the parties involved in the supply-chain. Companies do not have to commit their supply-chain partners anymore as much as they used to as the proposed process consists of a 'trustless' information system.

The blockchain used in this paper is 'Ethereum'. Further approaches would be to build up a whole new blockchain e.g. with the 'Hyperledger' framework. This would have gone beyond the scope of this paper, which is the reason, why this was not tested. The big network in the 'Ethereum' network makes it safe to use. The advantages of the big community in terms of this use case is debatable and it changes the cost structure of the information process as you do not pay anymore to the 'Ethereum' network but have to run your own blockchain. If this leads to cost advantages has to be worked out in a different paper.

References

1. Open Linked Data Discovery, Proof and Trusthttp://www.seoskeptic.com/open-linked-data-discovery-proof-and-trust/ , last accessed 2019/01/28.
2. Quote from Nick Szabo (1994), Pons, Jerome. (2018). Blockchains and smart contracts in the culture and entertainment business page 10.
3. Monika Solanki and Christopher Brewster (2013): Consuming Linked data in Supply Chains: Enabling data visibility via Linked Pedigrees
4. BLABEL Leaning and Labelling RDF Graphs https://blabel.github.io/ , last accessed 2019/01/28.
5. Linked Data Logistics https://git.scc.kit.edu/uidqr/linked-data-logistics , last accessed 2019/01/28.
6. Weihnachtsgeschäft Chaostage für Paketdienste, https://www.tagesschau.de/wirtschaft/pakete-105.html , last accessed 2019/01/28.