



## FoodFile

A DLT inspired food tracing system

Conceptualizing and discussing a non-blockchain solution

## Abstract

Food traceability across multi-step supply chains is a public interest and a legal requirement in countries worldwide. Today, most food producers maintain their records on paper or digital spreadsheets. Trace execution, which is essential to limit foodborne outbreaks, is a manual and slow process.

Recent research and implementations trend towards Distributed Ledger Technology (DLT) / Blockchain based solutions to facilitate recordkeeping and tracing. This work analyses the suitability of these technologies for the given domain by a deliberation of complexity and advantages. It is found that the general benefits of DLT and Blockchain do not fully apply in the context of food supply chains and hence do not balance out the added complexity.

A new, DLT inspired, model is proposed, implemented and evaluated by means of a simulation. The model is designed for bi-directional tracing in supply chains that span across multiple producers. Built as a decentralized approach, it allows for on-site data storage, granular information sharing and consumer facing information accessibly. It can maintain replications to prevent data loss, detect contradictions and verify consensus on information and audit trails. The simulation proves that the introduced concept can be implemented with technologies available today and that it satisfies all objectives defined in this work.

# Contents

<b>1</b>	<b>Introduction .....</b>	<b>4</b>
<b>2</b>	<b>Fundamentals .....</b>	<b>6</b>
2.1	<i>Food tracing .....</i>	6
2.2	<i>Distributed Ledger Technology (DLT) and Blockchain .....</i>	10
<b>3</b>	<b>Objectives and Methodology .....</b>	<b>16</b>
<b>4</b>	<b>State of the Art .....</b>	<b>17</b>
4.1	<i>Implementations and Standards .....</i>	17
4.2	<i>Related Work .....</i>	19
<b>5</b>	<b>Analysis .....</b>	<b>21</b>
5.1	<i>Requirements .....</i>	21
5.2	<i>Satisfaction of requirements by existing Concepts .....</i>	23
5.3	<i>Suitability of DLTs for food tracing systems .....</i>	24
<b>6</b>	<b>Concept .....</b>	<b>30</b>
6.1	<i>Decentralized Architecture .....</i>	30
6.2	<i>Unified data format: Atoms &amp; Entities .....</i>	30
6.3	<i>Membership Service .....</i>	33
6.4	<i>Tracing .....</i>	34
6.5	<i>Labels &amp; Supply Chain Linkage .....</i>	40
6.6	<i>Caching and Reconciliation .....</i>	41
6.7	<i>Sharing and Security .....</i>	42
6.8	<i>Client .....</i>	44
<b>7</b>	<b>Comparing Concept, Formal Model and Implementation .....</b>	<b>45</b>
7.1	<i>Core Objectives and on-site data storage .....</i>	45
7.2	<i>Information sharing regulation .....</i>	46
7.3	<i>Data Replication .....</i>	47
7.4	<i>Histogramphy resilience and conflict detection .....</i>	47
7.5	<i>Implementation Details .....</i>	47
<b>8</b>	<b>Evaluation .....</b>	<b>49</b>
8.1	<i>Knowledge base distribution .....</i>	49
8.2	<i>Implementation &amp; Deployment .....</i>	49
8.3	<i>Exemplary Supply Chain .....</i>	52
8.4	<i>Scenarios .....</i>	55
8.5	<i>Comparison against Ruiz-Garcia et al. ....</i>	84
<b>9</b>	<b>Conclusion .....</b>	<b>86</b>
9.1	<i>Outlook .....</i>	88
<b>10</b>	<b>References .....</b>	<b>89</b>

# 1 Introduction

In 2017, twelve European countries and Hongkong were affected by a food scandal caused by the pesticide Fipronil found in Eggs [1, 2]. When the problem was recognized by public authorities, the eggs were already processed into yolk for various usages, mayonnaise used for potato salads and delivered as hard cooked eggs to cafeterias. In June 2018, after a similar issue and a period of uncertainty which supermarkets were affected [3], a member of the German Bundestag called for a number-coding for egg-containing products [4].

This proposal relates to the general idea of food traceability which the united nations define “as the ability to discern, identify and follow the movement of a food or substance, intended to be or expected to be incorporated into a food, through all stages of production, processing and distribution.” Furthermore, a product recall is defined “as the action to remove food from the market at any stage of the food chain, including that possessed by consumers. [...]” [5, p. 4]

A supply chain wide food traceability system does not just enable precisely targeted recalls, but also quality improvements, fraud prevention, waste reduction and consumer information [6, 3f.]. An example is a potential online resource where the consumer may enter (or scan) an item code to see where the ingredients of a particular package originate from and whether it was recalled. Food waste can be reduced by enabling a precise calculation of remaining shelf life and informed decision making about freshness by producers and consumers.

The European Union [7] and the United States [8] have regulations in place that require each participant of the food supply chain to keep records about source and destination of ingredients and products. China’s Food Safety Law of 2015 intends the establishment of a digital food safety tracing system [9].

It is questionable whether the legal requirements in the EU and the US are sufficient. Backtracking an item in a supplier network without an interlinked record system takes an unpredictable amount of time: In a case study, Walmart needed seven days to trace a pack of sliced mangos back to the farm. The report states that “[w]hen an outbreak of a food-borne disease does happen, it can take days, if not weeks, to find its source” [10, p. 2] – not mentioning the additional time needed to find all products originating from that particular supplier up-chain. A consumer wanting to know whether an item was recalled needs to manually compare it to an online list such as provided by the RASFF<sup>1</sup> (EU) the FDA<sup>2</sup> (US) or the AQSIQ<sup>3</sup> (China). There is no opportunity to validate if an item was treated in compliance with quality standards like e.g. continuous cooling.

---

<sup>1</sup> Rapid Alert System for Food and Feed, available at (24.11.2019): [https://ec.europa.eu/food/safety/rasff\\_en](https://ec.europa.eu/food/safety/rasff_en)

<sup>2</sup> Food & Drug Administration, available at (24.11.2019): <https://www.fda.gov/safety/recalls-market-withdrawals-safety-alerts>

<sup>3</sup> General Administration of Quality Supervision, Inspection and Quarantine, only listing import/export products, available at (24.11.2019): <https://www.aqsiq.net/recall.htm>

From a technical perspective, most of today's food traceability systems are isolated solutions applied to supply chains where a limited set of parties decided to collaborate [11, p. 181]. Recent developments suggest Blockchain and distributed ledger technology (DLT) as basis for improvement [12, p. 4]. For example, IBM partnered with companies in the food industry<sup>4</sup> to introduce the permissioned and commercial platform 'FoodTrust'. It is based on Hyperledger Fabric, a DLT incorporating Blockchain technology designed by IBM and developed by the Linux Foundation [14, 15].

The following sections discuss characteristics, benefits and drawbacks of DLT usage with a focus on the food supply chain. The paper proceeds with a suggestion and an evaluation of an alternative approach.

---

<sup>4</sup> Dole, Kroger, Nestlé, Tyson Foods, Unilever, Walmart, Carrefour and others [13].

## 2 Fundamentals

The first part of this section serves as a brief introduction to the domain of food tracing, covering a definition, the legal background, advantages and practices today. The second part provides an overview of Distributed Ledger Technologies.

### 2.1 Food tracing

According to the definition of the United Nations (see 1), food tracing allows to follow food and food related goods individually along the supply chain downwards, relating to the trace to their origin(s), and upwards, relating to the trace to the consumer. Three major aspects can be identified for this process:

1. Labeling
2. Data acquisition
3. Data retention

Labeling addresses the requirement of making individual items uniquely identifiable. As related cost scales proportionally with production size, low-cost solutions are desired. Data acquisition relates to the question of

- a) what information is recorded: The chain of initial production and subsequent processing (merging) of goods, involved companies, coordinates, temperature and humidity information, quality/acceptance reports, etc.
- b) how information is recorded: Manually on paper, manual input to digital systems or automated by sensors and production equipment using IoT<sup>5</sup>-Technologies.

Data retention refers to the challenge of storing the captured information in a centralized or distributed manner considering issues such as integrity, consistency, versioning, access control, availability and fast accessibility. The solution described in this paper falls in the domain of data retention.

#### 2.1.1 Legal Background

Many countries worldwide have laws in place that set a minimum standard for food tracing. In 2002, the European Union released a regulation [7, Article 18] according to which

“[...] traceability of food, feed, food-producing animals, and any other substance intended to be, or expected to be, incorporated into a food or feed shall be established at all stages of production, processing and distribution.”

The regulation requires operators to be able to identify their suppliers and customers and make this information available to public authorities on demand. Furthermore, food which is placed on the market “shall be adequately labelled or identified to facilitate its traceability.”

---

<sup>5</sup> Internet of Things

A very similar approach is taken by the United States, where the Bioterrorism Act of 2002 states that

“[t]he Secretary, in consultation [...] with other Federal departments and agencies [...] may by regulation establish requirements [...] to allow the Secretary to identify the immediate previous sources and the immediate subsequent recipients of food, including its packaging [...]. [16, Sec 306 / 414]

In 2016, pursuant to the Food Safety Modernization Act, the FDA<sup>6</sup> reported to congress:

“Establishing a uniform set of data elements, a method of linking product along the supply chain, and advancing FDA’s ability to receive and analyze the data will provide the most basic and significant advancements and are a focus of FDA’s recommendations.” [17, p. 4]  
 “FDA believes that there are four major gaps in the existing recordkeeping requirements: 1. Lack of coverage of all establishments (e.g. farms and restaurants are excluded) [;] 2. Lack of uniform data and record requirements [;] 3. Inability to link incoming with outgoing product within a firm and from one point in the supply chain to the next [;] 3. Inadequate mechanisms to rapidly capture, receive, and analyze tracing information (electronic and technology applications).” [17, p. 10]

This expresses the conflict between the legal requirements and what would be practically relevant. China has recognized this problem and released the Food Safety Law of the People’s Republic of China (2015), setting the following objectives<sup>7</sup>:

“The state shall establish a whole process food safety tracing system. [...] The state shall encourage food producers and traders to gather and retain production and trade information by digital means to establish a food safety tracing system. The food and drug administrative department of the State Council shall establish a whole process food safety tracing cooperation mechanism [...]” [9, Article 42]

### 2.1.2 Advantages

An ideal cross-supply-chain food traceability solution allows for up- and downstream tracking. Multiple benefits can be identified for such a solution:

- Improved public health
- Increased customer confidence
- Waste reduction
- Cost savings

In case contaminated food is found at any stage of the supply chain, it can be traced to its production facilities and beyond to identify the true origin of the issue. In a second step, a downstream tracing can determine other affected products and their location. Precisely targeted recalls with a shortened response time lead to improved public health, because less people become affected of a potential foodborne outbreak [18].

---

<sup>6</sup> Food and Drug Administration

<sup>7</sup> translation by HFG Law & Intellectual Property

The availability of detailed information about product ingredient origins enables the consumer to make informed purchase decisions. This can benefit brand reputation and customer confidence [19, p. 56], reduce food fraud [20, p. 105] and opposes incorrect information, e.g. spreading on social media [21, p. 115].

Fast and precise traceability leads to “[r]educed food waste due to destruction of wholesome food in a category-wide recall.” [22, B25] Knowledge about storage, treatment and cold chain facilitate an improved prediction of remaining shelf-life [23]. While this itself can reduce food waste, it also enables a First-Expire-First-Out (FEFO) strategy [6, p. 398] or a First-In-First-Out (FIFO) strategy to reduce spoilage [19, p. 56].

The knowledge of product flows and supply chain visibility can improve inventory management and demand forecasting [19, p. 55]. As shown in several studies, direct cost savings arise from a reduction of delivery errors and quality claims [19, p. 57]. If data is available for individual products, automatic price markdowns at the point-of-sale (POS) are feasible [19, p. 56]. It can be expected that an improved recall capability lowers liability insurance costs [22, B25].

### 2.1.3 Methods

Several methods of labeling are available for food products. While IDs and relevant information can be printed on a tag and be attached to the product, the human-readable form should be accommodated with a digital equivalent such as: Barcode, QR-Code or Data-Matrix-Code. Several encodings exist for barcodes with most of them only supporting numerical signs. Since barcodes are one dimensional, they typically need more space for a given amount of information than QR- or Data-Matrix-Codes. On the other hand, scanners are simpler than those for QR-Codes and Data-Matrix-Codes. QR-Codes [24] and Data-Matrix-Codes [25] both support alphanumerical signs.

An alternative to printed tags are RFID<sup>8</sup> / NFC<sup>9</sup> chips. They do not require intervisibility, have a higher integrity in harsh environments and enable a higher reading rate than traditional barcodes. The technology is more expensive than printed barcodes and the allowed frequencies vary by country [6, 394ff.]. The capabilities of the chips are determined by the chosen standard and range from carrying an unmodifiable code to execution of cryptography tasks and other computations [26].

It is worth noting that biological ingredients contain DNA which can be extracted, analyzed and compared against databases such as BOLD<sup>10</sup>. As of today, this implicit labeling allows to infer the geographic origin in some domains, e.g. seafood [6, p. 396].

If product scans along the supply chain are exhaustive and properly logged, they reflect which product was in which production section at what time. To file the chain of processing steps

---

<sup>8</sup> Radio-Frequency Identification

<sup>9</sup> Near-Field Communication, a subcategory of Radio-Frequency Identification

<sup>10</sup> Barcode of Life Data Systems, see <http://barcodinglife.org>



and intermediate products a person or device with knowledge about the production flow is required in order to merge sensor- and scanner-data to meaningful records.

Regarding data retention, a research summit of the IFT<sup>11</sup>, summarized by [22], came to the conclusion that “[...] a single system that contains all traceability data is not considered a practical solution.” The report distinguishes between three approaches:

Approach 1:

“Each supply chain participant is responsible for collecting incoming shipment records from its suppliers, capturing any transformations within its own facilities, and retaining outgoing shipment records to its customers.” In this approach, a trace-back scenario remains a manual activity in the sense that “[...] regulators have to go to each node in the supply chain (consecutively)” which causes workload and delays.

Approach 2:

“[E]ach downstream recipient of a food product receives a record of the history of that product’s movement through the supply chain.” This architecture makes information quickly accessible but is paired with a “loss of confidentiality when revealing upstream trading partner information to downstream customers.”

Approach 3:

A trace “[...] begins with a general broadcast request (through some type of data query technology) [in order to] limit the investigation to only the most appropriate locations. Once these locations are identified, then deeper investigation occurs in a manner similar to Approach 1 [...]. The primary difference in the results between this approach and the status quo is that electronic record-keeping of traceability data and standards permits industry to discover and deliver information from a critical point with greatly improved efficiency. [...] [D]ue to alignment with current mandates, this approach can deliver a higher likelihood of early participation by the industry. [...] [I]f executed correctly, it preserves confidential trading relationships.”

The solution developed in this paper is a concretion of approach three. If and how blockchain and distributed ledger technology align with the presented scheme is discussed in section 5.3.

#### 2.1.4 Common Practice Today

Today, companies keep their records on paper, digital but manually managed or digital and automated. While it is difficult to quantify these practices, reports [20, p. 101][12, p. 6] indicate that manual work and handwritten records remain an important factor when conducting tracebacks.

A “lack of standard structure or format and the need to re-enter data” was identified by Bhatt et al. [27]; “The process of conducting a step-wise traceback (one supply chain node at a time) was complicated and oftentimes confusing. Most firms provided information in the form of PDF documents.” [27]

Accenture reports “[...] huge efforts in the tracking and reconciliation of data for a single transaction, from start to finish [...]”. [12, p. 6] Kennedy and McEntire characterize the absence of whole chain traceability by the fact that “data sharing is managed explicitly between two or

---

<sup>11</sup> Institute of Food Technologists, see <https://ift.org>

more parties. The challenge for the industry will be to agree on what traceability data is necessary to capture and store [...] and how to retain best data stewardship and governance practices.” [11, 181f]

## 2.2 Distributed Ledger Technology (DLT) and Blockchain

Distributed Ledger Technology has become relevant for the domain of food tracing. This section introduces the state of the art of DLTs to enable a suitability review (section 5.3).

### 2.2.1 History

In 1991, Haber and Stornetta described a cryptographically secured chain of blocks [28]. Nine years later, Konst presented an example implementation [29]. In 2009, the whitepaper of the cryptocurrency Bitcoin, which remains formative about the today's understanding for Blockchain technology, was released [30]. The concept of distributed ledger technology was first described by Lamport et al. in 1982 [31, p. 397] in conjunction with the Byzantine Generals Problem. An early implementation was proposed by Castro & Liskov in 2002 [32].

With the rise of Bitcoin, the idea of decentralized Byzantine-Fault-Tolerant (BFT) consensus (see 2.2.4) gained attention by several industries, especially the finance industry [33]. This development led to new DLT implementations, many of which do not comprise a cryptocurrency.

### 2.2.2 Distributed Ledger Technology

As there is no single recognized definition of Distributed Ledger Technology in literature [34, p. 15], this work is based on the definition by Beñčić and Žarko [35] referring to Narayanan and Clark [36]:

“Distributed ledger technology (DLT) enables the maintenance of a global and append only data structure by a set of mutually untrusted participants in a distributed environment. The most notable features of distributed ledgers (DLs) are immutability, resistance of censorship, decentralized maintenance, and elimination of the need for a trusted third party.”

The listed features are, what DLTs have in common with Blockchains and therefore many (but not all<sup>12</sup>) DLTs are built on top of Blockchain technology [38, p. 483]. DLT implementations achieve those features, except for immutability, leveraging decentralized BFT consensus mechanisms (see 2.2.4).

As the word ‘ledger’ suggests, the data structure of many DLTs is designed as a global state recording ownership of tokens. These tokens either represent a virtual value (in which case the DLT is a cryptocurrency), a conventional monetary value (e.g. Euro or Dollar) or a physical value (e.g. barrels of oil). How the representation of values relates to consensus is discussed

---

<sup>12</sup> E.g. Corda is a Distributed Ledger Technology which does not comprise a blockchain. [37].

in 5.3.2. Systems which record ownership use the term ‘transaction’ for state changes. Transactions must be validated to ensure that the issuing party owns the tokens it wants to spend [39, 8f]. We will use the term ‘holistic’ to describe this type of DLT. A more detailed explanation can be found in section 5.3.5.

### 2.2.3 Blockchain

Cong and He define a blockchain as a “distributed database that autonomously maintains a continuously growing list of public records in unit of ‘blocks’, secured from tampering and revision” [40, p. 1760].

Instead of being append-only (immutable) by agreement of all participants, Blockchains are append-only by design: Information to be saved is collected and bundled to a block which is attached to the existing chain of blocks of previous data. Every block has a hash value calculated based on its contents and the hash of the previous block. Hence, a change in any historic block would change its hash and propagate along the chain. By comparing the hash of the most recent block, two instances can verify that they have an exact copy of the data. [41, p. 7]

### 2.2.4 Consensus and Byzantine-Fault-Tolerance

The consensus problem requires agreement among several processes regarding shared data, while some of the processes may fail or be unreliable in other ways. Therefore, some degree of fault tolerance is required [42]. Algorithms such as Paxos (1998) [43] and Raft (2014) [44] solve the consensus problem as it is known from conventional distributed computing, but they are not resistant against Byzantine Faults. A common way to achieve resistance of censorship, decentralized maintenance and elimination of the need for a trusted third party in DLTs / Blockchains is the usage of a Byzantine-Fault-Tolerant (BFT) consensus mechanism.

The term BFT is related to the Byzantine Generals Problem which, in short, consists of several generals which need to agree on a shared information (an attack time). It is crucial that they agree on the same time because they may only win if they attack altogether. Finding consensus is hindered by the facts that

- they must communicate through the enemies’ camp, so the communication channel itself is neither secured nor reliable in terms of message delivery or speed.
- one or more generals may be a traitor, meaning that they can lie about their choice.

An algorithm is byzantine fault tolerant if it is resistant against the characteristics of the Byzantine Generals Problem as long as the number of traitors does not exceed a certain threshold [31]. BFT consensus mechanisms can be classified as follows (inspired by [41, 11ff]):

- Proof of Work (Pow)
- Proof of Authority (PoA)
- Proof of Stake (PoS)

- Proof of Space / Proof of Capacity
- Hashgraph
- Directed Acyclic Graph

#### 2.2.5 Proof of Work (PoW) based Consensus

In PoW, that the block hash needs to be smaller than a given number. To meet this requirement, the creator of the block can modify one field of the block header called Nonce. As hashing algorithms are one-way functions in a mathematical sense, it is impossible to compute the nonce based on the desired outcome. Instead, the creator must find a suitable nonce using brute-force. This process is called mining. Whenever a participant found a nonce (mined a block), s/he broadcasts it to the network. The block itself contains a fixed amount credited to the miner – the block reward. In addition, the miner collects the transaction fees for the transactions included in the block. All miners in the network switch their mining activities to be based on the recently attached block as the majority always works on the longest chain. By not switching to the longest chain, a miner risks the rejection of his/her next block and hence a loss of money (represented by the resources used for mining). Proper block validation is incentivized by the fact that other miners will reject a block containing invalid transactions: They will not risk to mine based on a block which is likely to be rejected by others. PoW is BFT. PoW is usually not used in permissioned environments as there is no justification for the immense energy consumption. Additionally, PoW based block creation requires time and implementations constrain the block size. This leads to a limited amount of transactions per time (e.g. Bitcoin with a fixed block size limit of 1MB and a creation rate of 10 minutes: 7 transactions per second). [41, 11f] [30]

#### 2.2.6 Voting / Proof of Authority (PoA) based Consensus

While the consensus in PoW implicitly originates from the fear of losing money (work/energy), voting based mechanisms reach consensus by agreement. A block is crafted and proposed by a node which has the privileges to do so. Depending on the implementation, this might be just one node which changes on a regular basis (after a certain number of blocks or after a certain timespan), a set of nodes with special permissions or any node. The block is added to the chain if and only if a (qualified) majority of nodes accept it. Depending on the implementation, not all nodes might have the right to vote for or against a block. Votes are digitally signed and (depending on the implementation) added to the block to proof its legitimacy. [41, p. 13] [45, p. 8] Voting based algorithms can be ‘truly’ decentralized, or leadership based (with Proof of Authority).

Examples for truly decentralized voting-based consensus are PBFT (Practical-BFT) or a variation of PBFT called Tendermint-BFT. As there is no leader, votes are broadcasted across all voters (nodes with the right to vote). As a result, network traffic increases exponentially with

the number of involved nodes, as every voter communicates its decision to all others. [45, p. 8] [46] [47, p. 5]

Leadership based voting algorithms are enabled by Proof of Authority: As an alternative to the high network traffic in PBFT (and its variations) and in order to be able to punish malicious behavior (e.g. approving invalid blocks), in PoA based consensus, a temporary leader is elected. This leader is responsible to propose a new block and coordinate the voting process for this block. To become eligible to be elected as a leader, a node (the owner of a node) must be a trustworthy (publicly) known entity which fears (legal) consequences in case of misbehavior. The leader changes on a regular basis, e.g. after each block, after a certain number of blocks, after a certain timespan or if the current leader is found to be malicious. [45, p. 8] [48, 3ff]

### 2.2.7 Proof of Stake (PoS), Space or Capacity based Consensus

Instead of PoW, where money is at stake through work / energy consumption, in PoS, money is at stake in the form of tokens of the currency of the chain which is using PoS. To be allowed to create a block, a participant is required to lock some tokens. The fact that a user wishes to lock tokens is stored on the chain and prevents the user from spending the amount. It does not change ownership if the participant does not behave in a malicious way (in which case the participant loses the tokens).

There exist multiple methods to decide which block is added next to the chain, who can propose a blocks and which paradigm the chain follows. Randomized block selection uses the hash of the last block in combination with the locked stake to determine the next block proposer. This deterministic algorithm can be executed by each node independently. As the hash of a block is influenceable by the block creator, in early implementations, a node which was given the power to propose blocks could take actions to preserve that power. Coin-age-based selection uses currency amount times holding period in combination to randomization to determine the proposer for the next block. The chain paradigm can be the same as in PoW (longest chain), or it can be the chain with the most accumulated coin age involved.

As there is no need to apply brute force on a nonce in order to find/mine a block, PoS is energy saving compared to PoW. The downside of this benefit is the “nothing at stake” problem: It is inexpensive to mine on contradicting forks simultaneously to gain mining rewards and double-spend money. Solutions for this issue exists, such as e.g. delegated proof of stake. [41, 12f] [45, 5ff]

Proof of Space / Proof of Capacity conceptually follows PoW. Instead of work, storage space is required to solve a given challenge. [41, 13f] [45, p. 7]

### 2.2.8 Hashgraph

Hashgraph is a patented byzantine fault tolerant consensus algorithm. While the algorithm can reach consensus on arbitrary data, the official implementation called Hedera Hashgraph comprises a cryptocurrency [49].

The network synchronizes by using a gossip protocol: Nodes constantly and randomly contact each other and exchange their latest knowledge about the current ledger status. When a node receives the current ledger status from another node, it creates an event. An event contains:

1. a timestamp
2. an optional payload (transactions<sup>13</sup>)
3. a hash to its own last event, representing its own view about the current state of the ledger
4. the hash of the last event of the sending node, representing the view of the sender about the current state of the ledger.

During creation of such an event, the receiver can add data it wants to capture on the ledger as payload. When a node is asked to deliver its current ledger-status, it delivers:

- the latest event which may contain a payload added by this node.
- a signature for this event

The receiving node verifies the signature and hashes of the received event. It then creates a new event using the computed hash from the received event as part 4 (see list above). Because every event is represented as a hash, and because every event point to its two parent events, the resulting structure is a directed acyclic graph of hashes, a Hashgraph.

As a result of this mechanism, each node has a list of events which allow the deduction of the event lists of all other participating nodes. The only exception of this are the most recent events which are still spreading through the network. This implies that each node can locally infer how any other node saw the ledger a few seconds ago. This is used to retrospectively confirm and order data (transactions) in a manner of virtual voting. Virtual Voting is a multi-step process in which each member of the network concludes, how each other member would have voted on the data (-ordering). It is substantial for virtual voting that each member knows the number of participants involved in the network. Hashgraph has a high throughput compared to other DLTs. However, transaction finality speed decreases the more nodes are involved. [49, p. 12] [45, p. 8] [50]

### 2.2.9 Tangle / Directed Acyclic Graph

Tangle is a concrete distributed ledger implementation by the German IOTA Foundation. It comes with a built-in currency and stores transactions in an append-only Directed Acyclic

---

<sup>13</sup> The term transaction does not necessarily refer to a transaction in a monetary sense but to a change of the current ledger state (a change of the data currently maintained on the ledger).

Graph<sup>14</sup>. Any transaction in the graph points to a configurable amount of previous transactions which were verified upon its creation.

A node wanting to submit a transaction must validate a configurable amount of already submitted transactions to do so. The Graph may contain conflicting transactions (double spends); However only one of those eventually gets a majority of approvals by other transactions. Proper transaction validation and achieving consensus in case of potential double spends is incentivized by the fact that a transaction which points to invalid transactions is unlikely to be validated / approved. As a node wants the transaction it is about to submit to remain valid,

- it has an interest in performing the validity check for the transactions it will refer to in the current transaction.
- it will, in case of a double spend, approve the transaction which is likely to gain the most approvals by others.

Transaction propagation is motivated by the desire to receive transaction information. A node which is 'too lazy' and does not propagate transactions to its neighbors will be dropped by them. Hence, such a node would not receive updates about network activity anymore and could not publish transactions. [45, 7f] [51]

---

<sup>14</sup> A blockchain matches, mathematically speaking, the definition of a directed acyclic graph. IOTA Tangle stores transactions in a tree with multiple branches.

### 3 Objectives and Methodology

The primary knowledge object and scientific objective of this work is a digital information system to support whole chain food tracing. Derivatively, the secondary knowledge object is a suitability analysis of Distributed Ledger Technology for such a solution.

The deliverables regarding the digital food tracing system are a well-defined concept and a Minimum Viable Product (MVP) implementation. The deliverable of the analysis is a qualitative suitability assessment. It is based on an existing definition of DLT and the identified requirements for the information system.

The structure of this work follows the three-step process of analysis (5), concept and MVP (sections 6 and 7) and evaluation (8). During the analysis phase, requirements were deducted from legal circumstances, existing implementations and existing research in the food tracing domain. These requirements were compared against existing scientific proposals and implementations, with a focus on DLT. The concept design was an iterative process of conceptual work, implementation work and abstracting and integrating learnings from the proceeding MVP development. The resulting concept was evaluated by a simulation leveraging the MVP and compared to a similar concept referenced by Ruiz-Garcia et al.



## 4 State of the Art

Food tracing is subject to industry efforts and research. The listed examples indicate that in recent years, researchers and software architects tend to incorporate Blockchain and Distributed Ledger Technology in their work. It has become a common belief that advantages of these technologies can also benefit the food industry [12] [67]: “The use cases for Blockchain in food traceability are nearly the same as those for general traceability initiatives, which is a primary reason it is so aggressively being pursued by many industry leaders.” [68, 136f.]

FoodTrust (4.1.3) and Ambrosus (4.1.4) are two recent DLT based implementations which are summarized below. Section 4.2.3 provides an overview of current scientific research with a focus on DLT related solutions.

### 4.1 Implementations and Standards

Many implementations exist to overcome the limitations of paper-based records. These include Optel, FoodTrust, Ambrosus, TE-Food, WaBi, Provenance, FoodLogiQ and others. The selection presented below aims to be a representative overview. It does not focus on category-specific approaches (e.g. only for meat, only for fruits, etc.) such as HarvestMark or BeefLedger.

#### 4.1.1 EPCIS

Electronic Product Code Information Services (EPCIS) is a standard developed by the GS1, the institution overseeing the issuance of Global Trade Item Numbers (GTIN)<sup>15</sup>. The first version was published in 2007.

The EPCIS specification specifies only a standard data sharing interface between applications that capture visibility event data and those that need access to it. It does not specify how the service operations or databases themselves should be implemented. [52, p. 9]

EPCIS is not limited to food and developed to enable the exchange of supply chain event data between different companies. Examples for events are transformations (consumption of entities to create one or more new entities) or localizations. EPCIS is designed for entity-level data but also supports product-level data exchange. Notwithstanding that EPCIS is often associated with RFID, it is not tied to any specific labeling technology and entities can be identified with existing standardized codes or arbitrary URIs. The running instances of the multiple im-

---

<sup>15</sup> GTIN, formally known as European Article Number (EAN) or Universal Product Code (UPC) is a widely accepted identification system for products (and services). The GTIN is the number which is encoded in product barcodes (see 2.1.4) to enable automatic identification, e.g. by register systems in supermarkets. The book and journal identification codes ISBN and ISSN are integrated in the GTIN system. The GTIN identifies products but not product entities, e.g. all copies of a book have the same ISBN. The Global Data Synchronization Network (GDSN) is a standard and network for registration and dissemination of GTINs and related information [51].

plementations of EPCIS form a decentralized network. Entity specific information is maintained by the participants themselves. Each company can configure its EPCIS server regarding the data it shares with partners or the entire network. [52]

#### 4.1.2 Optel

Optel Group is a provider of commercial traceability solutions for food, beverages, pharmaceuticals and cosmetics. Software is sold to customers and customized for their needs. This approach leads to whole-chain traceability for Optel's customer, given that the company has the power to motivate their suppliers to incorporate the Optel hard- and software into their production process. Marketing material provided by Optel's describes different use cases with an isolated solution for each of them, leading to the conclusion that there is no common platform for data exchange. Optel offers the possibility to add an individual QR-Code to the packaging, providing customers with access to a web resource visualizing the flow of the product through the supply chain. [53]

#### 4.1.3 FoodTrust

IBM FoodTrust is a commercial service to record the history of individual food items. It is advertised as a Blockchain solution "[e]nabl[ing] full transparency by digitizing transaction records and storing them in a decentralized and immutable manner, eliminating the opportunity for fraud across the food system." [14, p. 4] FoodTrust is based on Hyperledger Fabric where, according to the official whitepaper, "the chaining of blocks exist[s] only to make the integrity verification of the block sequence by peers more efficient." [15, p. 7] FoodTrust is exclusively hosted by IBM on IBM servers. The current API specification does not feature an option to access the Blockchain or block hashes. [54] FoodTrust links the information uploaded by participating enterprises and provides data and visualizations of product flows down to the item level, depending on the sharing permissions chosen by the uploader. [14]

#### 4.1.4 Ambrosus

Ambrosus is an open source solution to store food and drug tracing records. It is based on an Ethereum-Fork combined with IPFS<sup>16</sup> and comprises its own cryptocurrency called AMB (Amber-Token). To ensure immutability, "smart contracts related to the Ambrosus protocol will run on the Ambrosus Blockchain, which will be periodically copied to the Ethereum main network for further validation." [55, p. 20]

The core concept behind Ambrosus is to tie Amber-Tokens to physical goods. As physical goods travel through the supply chain, split up and transform into new products, these pro-

---

<sup>16</sup> InterPlanetary File System, an open-source distributed file system where nodes decide themselves what files they store. <https://ipfs.io/>

cesses are reflected as AMB-transactions between companies. The Tokens are linked to measurements captured by sensors along the chain. When a customer buys a food-product, s/he becomes the owner not just of the product but also of the associated tokens which s/he may sell back to food companies. [55]

The Ambrosus whitepaper does not clarify how the inherently mandatory assignment of (crypto-currency based) money contributes to recordkeeping beyond “incentiviz[ing] customers to purchase Ambrosus-tracked products.” [55, p. 19]

## 4.2 Related Work

Scientific research conducted on food traceability includes reports to federal authorities [56], case studies [57], workshops [22] and conceptual work. The sections below aim to give a representative overview of concepts related to the one presented in this work. Not mentioned are category specific solutions such as the one proposed by Thakur and Harburgh 2009 (grain) [58] or Chen et al. 2019 (pork) [59].

### 4.2.1 RFID and EPCIS related work

He et al. (2008) [60] suggest additional security features for EPCIS servers that allow for information sharing among dynamically formed communities along supply chains, based on the concept of circles of trust. Additionally He et al. (2009) [61] describe a „solution for Integrated Track and Trace in Supply Chain based on RFID & GPS“. The paper focuses on an EPCIS implementation that integrates positional data captured during food transport. The presented solution assumes the usage of combined RFID/GPS tags.

Li, Liu, Liu, Lai and Xu (2017) [62] describe a „IoT-based tracking and tracing platform for pre-packaged food supply chain[s]“. RFID tags are leveraged to record assembly and disassembly of prepackaged food batches and combines this information with geolocations and data from other sensors, such as temperature. The proposed platform comprises quality evaluation and customer warnings but no product transformations (no tracking for ingredients).

### 4.2.2 Ruiz-Garcia, Steinberger & Rothmund

Ruiz-Garcia, Steinberger and Rothmund (2008) [63] present a “model and prototype implementation for tracking and tracing agricultural batch products along the food chain”. They describe a concept similar to the one presented in this paper, following the main objective to limit each company’s efforts of recordkeeping to what is required by law. The idea is that every company hosts a web-service which can reply to queries regarding product batches. Each web-service is connected to a local database containing information about suppliers, transformations, customers and geolocations of batches. The reply of a web-service is in a standardized format, allowing the services to automatically interact with each other and form a decentralized system. In the described implementation, a client looking for information about a

given batch executes a search by iteratively querying web services of different companies based on the acquired knowledge from previous replies. E.g. if a company A knows that a given batch was purchased from supplier B, a request to supplier B might show that it was the result of a transformation involving batches from suppliers C and D. The search will follow up to query supplier C and D about their batches.

For a detailed comparison of the approach suggested in this paper and the one by Ruiz-Garcia et al. see section 8.5.

#### 4.2.3 Blockchain based approaches

Tian (2017) [64] describes a high-level concept using a technology called BigchainDB to facilitate food tracing. The paper is built on the foundation that BigchainDB “combines the key benefits of distributed DBs and blockchains.” [64, p. 4] BigchainDB is a decentralized database / Blockchain storing data in the form of transactions. Voting is used to achieve consensus among participating nodes, whereas all nodes vote for each block. The validity of the block is determined based on whether or not a total majority of nodes voted for or against a given block. [65] The paper does not explain how the BigchainDB can be utilized in detail. For example, it remains unclear how the various processes of the food supply chain should be mapped to transactions and whether every participating company should host their own BigchainDB node. The consensus approach used by BigchainDB is known for poor performance in setups with large numbers of nodes due to the exponentially increasing amount of network traffic caused by broadcasting votes<sup>17</sup> (see 2.2.6). The paper neither mentions data formats nor how a search could be carried out through the network in order to assemble a product history containing transformations.

Hao, Mao et al. (2018) [57] published an “[i]nnovative Blockchain-Based Approach for [a] Sustainable and Credible Environment in Food Trade”. The paper describes a Blockchain based trading platform tailored for the specifics of food. The integrated auction mechanism considers the impact of transit time on food quality and value loss and factors in that multiple suppliers may be required to meet a given demand. The Blockchain consensus is voting based and does not introduce a currency. While the paper is an application of Blockchain technology in the food supply chain, it is not related to food traceability in the sense of batch-identification and recording a transformation history.

---

<sup>17</sup> The BigchainDB whitepaper presents a scalability testing with up to 32 nodes [65, p. 44]. In a newer version of the BigchainDB whitepaper released in 2018 (BigchainDB 2.0), Tendermint is replacing the proprietary consensus mechanism suggested in the referenced version from 2016 [66].

## 5 Analysis

Following the objective of this work described in section 3, requirements for a “digital information system to support whole chain food tracing” are identified in section 5.1. A comparison of the presented existing approaches against these requirements can be found in 5.2 and a suitability analysis of DLT based approaches in general is the subject of section 5.3.

### 5.1 Requirements

Table 1 categorizes the chosen requirements in core objectives, privacy and trust.

Core Objectives	Privacy	Trust
1. Whole chain traceability for individual entities 2. Bidirectional tracing	3. On-Site data storage, if desired 4. Information sharing regulation for individual pieces of information regarding specific entities 5. Information accessibility for non-participants (e.g. consumers)	6. Replication to the degree desired by the participants 7. Contradicting Information is detectable 8. Resilient histography of data changes

Table 1: FoodFile objectives

The DLT inspiration is apparent in the trust category. As expounded in 2.2 and further explained in 5.3.4, replication is a core feature of DLTs. This work does not strive for full data replication on all nodes as this is not expected to be possible considering the vast amount of data emitted by the countless food supply chains worldwide.

Objective (7) is close to what DLTs achieve with consensus mechanisms (see 2.2.4). It differs in that the resolving of potential conflicts is out of scope in this work. The reason for this decision is inherent to what consensus mechanisms were designed for and illuminated in section 5.3.2.

Objective (8) focuses on histography and resilience rather than ‘immutability’, which is the common term in the DLT domain. Except for Proof-of-Work driven DLTs that are hosted by mutually independent parties, DLTs generally achieve ‘immutability’ by distributing data among participants and providing automated comparison. As data is technically not immutable in these cases (detailed elaboration in section 5.3.3), ‘resilience’ is a more appropriate wording. Histography (also called auditing) is a desired side effect of resilience in most DLT implementations (5.3.3) and hence explicitly mentioned here.

The following sections provide reasoning for the chosen of objectives.

#### 5.1.1 Core Objectives

The two core objectives express the purpose of the system: The proposed solution should be able to store production, transformation and transportation records for individual products for the supply chain. This includes cases in which the flow of goods is not known beforehand, e.g. if batches are sold in an auction.

Whole chain traceability refers to the capability to follow a product through its entire supply chain, end-to-end and across all involved companies. As a single company may cover multiple steps of the supply chain for a product, a side effect of this objective is a partial overlap with the responsibilities of an inhouse traceability solution. In case of incomplete information, e.g. a missing transport between two locations, a missing product description or an incomplete list of ingredients, the system must still be able to retrieve and display all data it knows.

The approach must be able to perform tracing upstream, i.e. from the shelf to the ingredients, as well as downstream, i.e. from a farm to the retailer of the produced good. This bidirectional trace should work from every possible perspective of the supply chain. A trace from the perspective of an intermediate supply chain member researching about a self-produced entity looks different than a trace from a slaughter or retailer regarding their entities: A batch of meat might end up in many parallel intermediate steps – all of which are of interest for the slaughter – but a given intermediate producer might only care about the follow up steps to his production.

#### 5.1.2 Privacy

Resulting from objectives (1) and (2), the system will maintain information which uncovers relationships between businesses. As supplier- and customer data is business critical, every participant should have the choice with whom to share which information. Whole chain traceability will require some data to be accessible, yet the decision to use the technology must not implicitly enforce any opt-in to data distribution for three reasons: First, reservations can be mitigated if data ownership and access-control remain at the data originator. This contributes to a high adoption rate. Second, participants can negotiate their relationship-visibility individually, removing the necessity of an a priori legal compliance assessment. Third, it allows for a risk-free integration of the technology in an existing IT-landscape, simplifying the execution of an exploration phase. This not only explains the necessity of an access control mechanism, it also justifies the objective of enabling on-site data storage. In scenarios where cloud-based hosting is enforced by design, the security of data is dependent on the given solution architecture and/or trust. On-site data storage does not prevent third parties to offer a given solution as a managed cloud-based service. Consequently, objective (3) does not interfere with

the possibility to provide the solution as SaaS<sup>18</sup>. Even in areas where regulations enforce any type of data sharing, the responsibility of enforcing these regulations should remain at judiciary and executive, not at a technology.

Information accessibility for non-participants (5), specifically consumers, is essential to achieve an increased customer confidence, a key benefit described in 2.1.2. This objective is subordinated to (4) – it should remain the company's (or the legislative's) choice which data is publicly available.

### 5.1.3 Trust

The term trust embraces two aspects: The data provided by the system must be trustworthy and the system as such must be trusted to be effective and reliable.

Replication prevents data loss in cases of hardware failure or hardware destruction due to external events. In addition to this, objective (6) implies replication of shared information across the participants to the degree the participants are willing to allocate memory for replicated facts. A participant's system should be able to maintain copies of information it has access to. This prevents data loss in cases of bankruptcy or discontinuance of business.

The quality of data in a record-keeping system depends on the participants entering it. The role of the technology is to make contradicting information detectable (7). A mechanism to identify potential errors and manipulation is the foundation for a (partly) automated reconciliation<sup>19</sup>, which is out of scope of this work.

Correctness of data is essential, but errors in recording information happen occasionally and the ability to modify data is a necessity. To impede manipulation, all data changes must be audited (8). The replication requirement described above also affects the resulting audit-trails, fostering resilience through comparability analog to objective (7).

## 5.2 **Satisfaction of requirements by existing Concepts**

Before conceptualizing a new solution, sufficiency of existing implementations and research it is yet to be reviewed. Table 2 summarizes what the respective paragraphs of section 5.1 mention regarding the objectives of this work. A matching solution could not be identified.

---

<sup>18</sup> Software-as-a-Service

<sup>19</sup> E.g. dashboards raising alerts on contradictions to allow for human resolvment or automatic alignment of minor data differences.

Implementation or re-search concept	Suitability estimation
EPCIS	The EPCIS standard sets the foundation for supply chain tracing but an open source implementation with these capabilities could not be found. The standard does not specify how replication of data can be achieved and how contradicting information should be handled.
Optel	Optel builds tailored tracing solutions for their customers. It is not clear if they offer on-site data storage for each participant individually. From the information publicly available it can be concluded that the individual solutions cannot communicate with each other ad-hoc.
FoodTrust	FoodTrust is not decentralized and exclusively hosted by IBM.
Ambrosus	Ambrosus is an Ethereum fork and as such requires a cryptocurrency. The benefits of incorporating Blockchain in supply chain tracing are discussed in 5.3. Apart from these potential benefits (if any), the significant amount of added complexity by a cryptocurrency does not have any positive effects on the tracing capabilities.
Ruiz-Garcia, Steinberger and Rothmund (2008)	See 8.5
He et al. (2008)	He et al. do not cover product transformations and hence ingredients cannot be traced.
Tian (2017)	The BigchainDB based concept by Tian is not detailed enough to be checked against the objectives of this work. The technological implications of BigchainDB indicate that it will not scale to support many supply chain members.
Hao, Mao et al. (2018)	Hao, Mao et al. tackle a different problem and present a marketplace, not a tracing solution.

*Table 2: objectives & existing concepts*

### 5.3 Suitability of DLTs for food tracing systems

Distributed Ledger Technology is one potential option to store information. The three trust-objectives listed in Table 1 intersect with the definition of DLT (2.2.2) by Bečić and Žarko. Fault tolerant consensus mechanisms are an integral part of DLTs. Their purpose is to eliminate contradictions and they always comprise a certain degree of replication (see 2.2) DLTs maintain a resilient history due to their append-only design (for further explanation see 5.3.3).



Scientific research and industry in the field of food tracing are shifting attention towards DLT and Blockchain technology (see section 4). Considering the recent trends and the referenced properties of DLT, it is necessary to review the suitability of DLTs for food tracing systems.

### 5.3.1 Complexity introduced by DLT

The common technical approach to achieve resistance of censorship (2.2.2) is data reconciliation based on a Byzantine Fault Tolerant consensus mechanism (2.2.4). In traditional consensus such as Paxos or Raft a leader node serves as single point of truth for the alignment process. Due to the nature of the BFT requirement, the process of coordinating the nodes is more complex as the descriptions in 2.2.5 - 2.2.9 may indicate at a first glance. The complexity added by BFT-Consensus manifests in one of the following:

- Large energy / other resources consumption (Proof of Work, -Space or -Capacity)
- The need for a ledger inherent currency (Proof of Stake)
- Increased communication effort (Proof of Authority / voting, Hashgraph, DAG)

The append-only storage paradigm of DLTs is not ideal for fast data accessibility and not appropriate for any type of query. In DLTs such as Hyperledger Fabric [15], Bitcoin [30] or IOTA Tangle (2.2.9), the current ledger state is a logical result from the changes ('transactions') applied since the ledger was created. This concept works well in a cryptocurrency use case with non-negative account balances: In order to approve a desired transaction, one can go back in history, offsetting spending and earnings of that user, until the difference exceeds or equals the desired amount<sup>20</sup>. This approach is inappropriate for arbitrary data (binary, text, etc.) for which the common query demands the latest state. To answer such a query, it would be necessary to re-compute all changes since the origin of the ledger to deduct the most recent version. The common approach to overcome this drawback is to maintain the current ledger state in a separated data-structure: For example, Hyperledger Fabric, Ethereum [72, p. 3] and Hedera Hashgraph [49, p. 31] do this leveraging Merkle-Trees, Key-Value stores or both. In summary, the added complexity of the append-only storage paradigm used by DLTs arises from either

- the parallel maintenance of two storage paradigms, or
- decreased access performance due to the necessity of a deduction process

In the food tracing domain, additional complexity is added by ledgers which do not provide the participant with an option to choose what information to persist based on the content. If a ledger does not offer a fine-grained persistence regulation, a participant would be forced to either store large amounts of irrelevant data (from the participants perspective) or nothing.

---

<sup>20</sup> This technique is referred to as UTXO (Unspent Transaction Output) [41, p. 18]. Bitcoin is one of many cryptocurrencies using this procedure to validate transactions.

### 5.3.2 The benefit of resistance of censorship

Censorship resistance is “[t]he probability that a transaction in a DLT design will be intentionally aborted by a third party or processed with malicious modifications.” [73] All presented Byzantine Fault Tolerant consensus mechanisms which achieve this censorship resistance do so by obtaining a majority (2.2.5 - 2.2.9). This feature is essential in scenarios, where the data records on which consensus is achieved inherently represent the truth. The most common scenario for this is are UTXO cryptocurrencies: The wealth of each participant is determined exclusively by the transactions made from and to the respective account. In other words, *the recorded data becomes the financial truth for the participants through the process of achieving consensus*.

This is not applicable for scenarios where an existing truth is to be recorded retrospectively. Examples for events or states of potential recordkeeping interest are a legal transfer of goods<sup>21</sup>, a processing or a geolocation of intermediate products or a state such as a temperature. Only a human or a sensor with direct perceptive access can produce a valid descriptive record. *The truth of such a record can neither be validated by an uninvolved party, nor is its correctness a matter of majorities*.

In context of consensus mechanisms and recordkeeping, ‘data validity’ does not refer to the truth of records; it refers to the majority agreement on what information to persist. As opposed to cryptocurrencies, in the food tracing domain, these two aspects are not synonymous. In the food tracing domain, the resistance of censorship does not advance data quality or correctness but achieves consistent data redundancy. The reason is that a shared agreement requires the data to be distributed accordingly either beforehand or implicitly.

### 5.3.3 The benefit of Immutability

Before illuminating the benefit of immutability claimed by DLTs, two commonly desired properties need to be described and differentiated:

- The immutability of data persistence (here abbreviated as IDP) and
- the mutability of information.

In DLTs, IDP is achieved through the combination of two aspects:

- “Once data has been recorded in the ledger, it cannot be secretly altered expost without letting the network know it (data is tamper-resistant).” [41, p. 7]
- The respective DLT implementations consider altering data as forbidden. Because (and only if) the nodes of the DLT are controlled by independent legal entities, one can trust that there is no common interest to change this shared agreement.

It can be argued that in Blockchain-based DLTs, immutability is not just backed by a shared agreement not to change any data, but also by the fact that a fork would be required to do so.

---

<sup>21</sup> or even other not exclusively ledger managed values such as Euro / Dollar

However, if a majority of the consensus drivers decide to change historic data, they can re-apply the consensus-mechanism to the blocks recorded after the given change: “Immutability is relative and relates to how hard the history of transactions is to change. Indeed, it becomes very difficult for an individual or any group of individuals to tamper with the ledger, unless these individuals control the majority of ‘voters’.” [41, p. 7] The only consensus mechanism which makes altering a time-consuming effort even for majorities is POW (2.3.5), where the speed of computing a successful fork is proportional to the size of the majority.

The mutability of information takes into account that records occasionally contain errors resulting from human mistakes or technical failures of the data source. Analog to the reasoning from the previous paragraph, use-cases where the truth is recorded retrospectively (such as food tracing) and not inheritably achieved by the consensus mechanism (such as cryptocurrencies), required data to be modifiable.

Mutability of information in conjunction with IDP can be achieved with data versioning, also called auditing: When a given information needs to be changed, instead of modifying the persisted data, a new version is stored (appended) to the ledger. While this is a suitable approach to couple IDP with information mutability, auditing per-se is an established technique with many usage scenarios (E.g. data warehousing) and does not require IDP.

When weighting the benefit of immutable data persistence offered by DLTs for the food tracing domain, the key considerations are:

- As mentioned before, the data must be hosted by mutually independent parties. If non-participants should be given the power to validate that a certain information was not altered, they must have sufficient read access to recompute and validate all reconciliation critical data, e.g. a merkle-tree or a hash-chain.
- While auditing can be achieved without IDP, it adds the benefit conclusiveness.
- The correctness of information is not improved by IDP, because the question whether the most recent version of a record reflects the truth cannot be deducted from its modification history.
- The provable frequency in which participants change their data records may serve as an indicator for their credibility.

#### 5.3.4 The benefit of decentralization

Decentralized maintenance and the elimination of the need for a trusted third party are properties of the consensus mechanism which is used by the DLT. Therefore, those benefits apply if

- the chosen consensus mechanism is not leadership-based, as for example Raft is. A system with pluggable consensus, such as Hyperledger-Fabric, loses this property when used with a leadership-based consensus.

- the data is hosted by mutually independent parties. If not, the single or the few hosting parties implicitly become the trusted party (parties).
- The consensus drivers, which dependent on the DLT are called voter nodes, mining nodes, consenter nodes etc., are equally distributed among the independent hosting parties. Otherwise, the single or the few hosting parties of consensus drivers implicitly become the trusted party (parties).

Many existing DLT implementations, especially enterprise DLTs, store data in a distributed manner but comprise a centralized lookup service for members. The responsibilities of such a service are comparable to those of a domain name registrar in conjunction with a domain name service. Examples are Corda [37] or Hyperledger Fabric.

A central membership service, if applicable, is acceptable in the food-tracing domain if it is transparent in its behavior and either governed by a public trustworthy authority or easily replaceable in case malicious behavior is detected. Transparency requires equal and justified acceptance criteria for applying members, equal treatment of members and a correct carryout of lookup services.

Decentralized maintenance and the elimination of the need for a trusted third-party ties in with the requirement for optional on-site data storage (5.1.2) and trust (5.1.3). The benefit contributes to a reliable architecture, since it removes a potential single-point-of-failure. It may also increase the trust of participants in the system as such.

### 5.3.5 The benefit of a holistic representation

The term ‘transaction’ in the context of DLT reflects a change to the latest version of the stored data.

As discussed in 2.2.2, many DLTs are designed to store information as a global state (‘ledger’). There, the term ‘transaction’ specifically records a token exchange between participants (which always involves a state change). This approach originally represented the use-case of cryptocurrencies and is not suitable for any type of information. The Ambrosus implementation (4.1.4) transfers this concept to the food tracing domain where the participants are enterprises, tokens represent food items and transactions reflect their processing along the supply chain.

DLTs which are capable to handle token based transactions and prevent double-spending could potentially be used to create a holistic record of the food market: Every ownership-change and every processing would be logged as a transaction which together form the current ledger state, analog to a cryptocurrency. Harvested food or bred animals would have to be registered on the ledger by certified authorities<sup>22</sup>.

---

<sup>22</sup> inspired by Corda, where non-cryptocurrencies can be ‘put on ledger’ using certified authorities, e.g. banks [37].

For this work, such an approach is out of scope because advantages and drawbacks are considered not to be in balance: The benefit is neither the holistic record, nor it's correctness but just it's consistency. Transaction logging can be achieved in most general-purpose storage mechanisms, e.g. SQL- or NoSQL-Databases. While the described DLT usage would guarantee consistency among the transactions, it cannot ensure their correctness (see 5.3.2). The advantage of guaranteed consistency comes at the cost of introducing authorities and bureaucracy for entity registration and conflict resolving. This is a considerable drawback in terms of cost efficiency and an industry wide rollout.

### 5.3.6 Conclusion

At a first glance, Distributed Ledger Technology looks applicable to underpin a food-tracking system. It was shown that DLT is accompanied by a high technological complexity needed to achieve the claimed benefits. The complexity arises primarily from the modalities of

- node coordination
- efficient data access

An analysis of DLT advantages in the context of food-tracing yielded the following insights:

- Resistance of censorship does not advance data quality or correctness but achieves consistent data redundancy.
- Immutability adds the benefit of conclusiveness to auditing, but auditing can be achieved without immutable data persistence.
- If hosted by mutually independent legal entities among which consensus participation is equally distributed, DLT facilitates a desirable manner of decentralization.

Balancing these results with the complexity introduced by DLT, this work concludes that, for the domain of food traceability, an improved combination of simplicity, functionality, privacy and trustworthiness are likely to be achievable. The following sections present a concept design and an MVP for this objective.

## 6 Concept

The presented concept is at its core a composition of a unified data format, a membership service and a trace algorithm. These core aspects are formalized in the sections 6.2 - 6.4. This model sets the foundation to meet the privacy and trust requirements with the corresponding approaches being described in 6.5 - 6.8.

As groundwork for the upcoming formalization, we define function  $t$  for accessing specific elements of ordered tuples.

Let  $Y = (h_1, h_2, h_3, \dots)$  be a tuple of type  $Y$ .

Let  $Z = (h_1', h_2', h_3', \dots)$  be an instance of a tuple of type  $Z$ .

We define  $t$  as the function returning the specified element of the specified tuple-type from the given tuple:

$$t_{h_1}^Y(Z) = h_1'; t_{h_2}^Y(Z) = h_2'; t_{h_3}^Y(Z) = h_3'; \dots$$

We denote  $P$  as the function returning the power set of a given set.

### 6.1 Decentralized Architecture

FoodFile is decentral. Each supply chain member can store and host data using its own FoodFile deployment<sup>23</sup>. Conceptually, each FoodFile server exposes two Interfaces: One interface to connect to FoodFile instances of other members and one interface to communicate with internal services such as a frontend for data retrieval and production systems for data supply.

### 6.2 Unified data format: Atoms & Entities

FoodFile stores information on the level of physical entities. A physical entity is an individual, concrete product. As physical entities travel along the supply chain, they undergo ownership changes and further processing to new entities. The producer of a physical entity assigns an ID for it and attaches this ID to the product using some sort of label (2.1.3). Downstream supply chain members scan this ID as part of their processes and store their individual knowledge about a physical entity on their FoodFile system. Within the scope of this concept, the term entity is used for this virtual representation of a physical entity. Entity records are maintained in a granular and merging optimized manner: A single information about a physical entity, such as a transfer or a processing, is stored in an encapsulated format herein called information atom. There are several types of atoms for the different types of information to store. What all atoms have in common is

---

<sup>23</sup> If desired, such an instance can be deployed to the cloud or shared with others. SaaS deployments are possible.

- a field for the entity-ID to which the information belongs. Entity-IDs are Globally Unique Identifiers (GUIDs).
- an ID field for the atom itself. The ID is a random string because it needs to be generated on an individual FoodFile instance without communication to others. It is expected to be unique for the given entity.
- a version-field which is raised with every change. As updates can be applied without communication to others, the field is initially set to the UNIX timestamp marking the creation of this atom version. The exact value may be adjusted by reconciliation processes afterwards which is why this field is not labeled timestamp.
- a signature list, where each signature comprises an identifier for the issuing member, the signature value and a timestamp (see 6.6).
- a re-sharing allowed flag. See 6.7.4 for further information.

An atom  $A$  is defined as

$A = (a, e, v, I, S)$ , with

$a$ , the atom identifier

$e$ , the entity identifier

$v$ , the atom version

$I$ , the atom information

$S$ , a set of signatures

We define  $\ddot{A}$ , the universe of all atoms.

The re-sharing allowed flag is practically relevant as data security and data sharing are important topics for enterprise scenarios. It is not needed to describe the conceptual approach of FoodFile and the tracing mechanism, which is why it is excluded from the formal model notation.

The structure of  $I$ , the atom information, is determined by the atom type:

$$I \in \{I_C, I_T, I_D, I_I, I_R, I_E\}$$

The different atom types and their respective structures of  $I$  are described in sections 6.2.1 - 6.2.6. The atom information is unique for the combination of atom identifier, entity identifier and atom version:

$$\begin{aligned} \forall (A_1, A_2) \in \ddot{A}^2 \left( (t_a^A(A_1), t_e^A(A_1), t_v^A(A_1)) = (t_a^A(A_2), t_e^A(A_2), t_v^A(A_2))) \right) \\ \Rightarrow t_I^A(A_1) = t_I^A(A_2) \end{aligned}$$

FoodFile treats an information atom as the smallest indivisible unit of data in terms of sharing, auditing, integrity, validity and reconciliation. Both interfaces of a FoodFile server mentioned above expose data exclusively as lists of information atoms. Based on the definition of atoms, we define an entity as a set of atoms all describing the same concrete physical item:

$$E = \{A_1, A_2, A_3, \dots\}, \forall (B_1, B_2) \in E^2 \ t_e^A(B_1) = t_e^A(B_2)$$

We denote  $\ddot{E}$  as the universe of all entities and  $\ddot{D}$  as the universe of all entity-IDs.

We define the function  $i$  to return the entity identifier all atoms of the same entity have in common:

$$i: \ddot{E} \rightarrow \ddot{D}$$

$$i(G) = t_e^A(B) \text{ with } B \in G$$

### 6.2.1 Creation-Atom ( $I_C$ )

A creation atom logs the creation of a physical entity. It comprises a timestamp, a location, the responsible member and a list of entities consumed during the production. If the creation represents harvesting or animal-birth, the list of inbound entities is empty.

$$I_C = (C, r, l, t), \text{ with}$$

$$C \in P(\ddot{D}),$$

a set of identifiers for the entities consumed during production

$$r \in \ddot{R}, \text{ see 6.3}$$

$$l, \text{ a location}$$

$$t, \text{ a time}$$

We define the optional notation  $A^C$  to mark Creation-Atoms:

$$\text{We write } A^C \text{ for Atom } A \Leftrightarrow t_I^A(A) = I_C$$

$$\text{Let } \ddot{A}^C \text{ be the universe of all creation atoms.}$$

We define

$$c: \ddot{A}^C \rightarrow \ddot{D}$$

$$c(B) = t_c^I(t_I^A(B)),$$

a function that extracts the consumed entities recorded in the given creation atom.

### 6.2.2 Transport-Atom ( $I_T$ )

A transport atom logs the geological movements of a physical entity. It is a list of geo-track-points and a responsible member:

$$I_T = (p, r), \text{ with}$$

$$p = (l_1, l_2, l_3, \dots), \text{ a tuple (a list) of } n \text{ locations}$$

$$r \in \ddot{R}, \text{ see 6.3}$$

### 6.2.3 Description-Atom ( $I_D$ )

A description atom stores attributes about the Product and the quantity, e.g. *Fragaria Vesca* (wild strawberries); 5kg:

$$I_D \triangleq \text{a description text}$$



#### 6.2.4 Involvement-Atom ( $I_V$ )

An involvement atom represents the information that the referenced member was directly or indirectly involved in processing of the physical entity. This information may be recorded by members when they receive or ship entities in order to store the seller/buyer. During a trace this atom serves as a link between members of the FoodFile network; It points out other members who may provide additional information regarding the physical entity. This is complementary to the ‘responsible member’-fields in other atom types.

$$I_V \in \ddot{R}, \text{ see 6.3}$$

#### 6.2.5 Destruction-Atom ( $I_R$ )

A destruction-atom marks the destruction of a physical entity mentioning a location, a responsible member and a reason:

$$\begin{aligned} I_R &= (w, l, r), \text{ with} \\ w &\in \{\text{consumed}, \text{lost}, \text{wasted}, \text{unspecified}\} \\ l, &\text{ a location} \\ r &\in \ddot{R}, \text{ see 6.3} \end{aligned}$$

#### 6.2.6 Erase-Atom ( $I_E$ )

An example use case for deleting an atom is a sensor producing wrong measurements with the defect being recognized after persisting and sharing data.

$$I_E \triangleq \emptyset$$

Information is never deleted from FoodFile, but creating an atom copy with an increased version number and the atom information set to  $\emptyset$  has the semantic effect of a deletion.

### 6.3 Membership Service

To enable autonomous communication between FoodFile instances across multiple companies, the servers must know how to connect. The membership service is a publicly available register, mapping FoodFile member-IDs to company names and the respective FoodFile API endpoints. Participants decide which membership service they trust and configure their FoodFile instance accordingly. To simplify the formal model, we assume that there is only one membership service. We define Member  $M$  as

$$\begin{aligned} M &= (r, q), \text{ with} \\ r, &\text{ a member identifier} \\ q, &\text{ a contact information for the FoodFile instance of that member (URL)} \end{aligned}$$

We define  $\ddot{M}$ , the universe of all members

$$\ddot{M} = \{M_1, M_2, M_3, \dots\}, \text{ with } \bigvee_{(H_1, H_2) \in \ddot{M}^2} t_r^M(H_1) \neq t_r^M(H_2)$$

We denote the universe of all member-IDs as  $\ddot{R}$ .

A membership service provides public access to  $\ddot{M}$  by offering a search function:

$$f: \ddot{R} \rightarrow \ddot{M} \\ f(g) = (H \in \ddot{M}: t_r^M(H) = g)$$

A membership service does not only provide company information and an API endpoint for a given member-ID; it also serves as a certificate authority for X.509 certificates<sup>24</sup>. Certificates are issued for members and used to sign and validate information atoms. A legitimate way to operate a membership service is to allow signing up for everyone but conduct a company accreditation before issuing a certificate. This approach lowers the entrance barrier and accelerates the setup of instances. Already emitted information atoms may be signed retrospectively once a company decides to opt in for accreditation. Clients decide themselves how they display unsigned information atoms. This is further discussed in 6.7.

## 6.4 Tracing

When a client reaches out to its FoodFile server to trace a physical entity, it provides the entity-ID, the desired trace direction (upstream or downstream) and, optionally, an ID of a member which is likely to recognize the entity. The purpose of this optional parameter is explained in 6.5. The FoodFile server realizes tracing with a combination of a local and global recursive search. We define

$\ddot{A}_M$ , the set of atoms known by member M.

### 6.4.1 Local Search

As a first step of a trace, the server queries its own database for information atoms about the given entity-ID. The retrieved atoms are added to an in-memory knowledge base. In case of an upstream search, the result is filtered for a creation atom and the search is repeated for all inbound entities used for the creation of this physical entity, if any. In case of a downstream search, the database is queried for creation atoms which have this entity-ID as inbound. The search is then repeated for the outbound entity of this creation atom.

This process bottoms out when it does not deliver any additional atoms. At that point in time, all locally available information atoms directly or indirectly related to the provided entity are in the knowledge base.

---

<sup>24</sup> See RFC 4158 for further information about X.509 and Public Key Infrastructure.

We define the local downstream search as a function:

$$\begin{aligned}
 l_d: \ddot{D} \times \ddot{M} &\rightarrow P(\ddot{E}) \\
 l_d(i, H) &= \begin{cases} \emptyset, & \text{if } \nexists B \in \ddot{A}_M: t_e^A(B) = i \\ \{G_1, G_2, G_3, \dots\}, & \text{else, with:} \end{cases} \\
 &\quad t_e^A(B) = i(G_n) \Rightarrow B \in G_n, \forall B \in \ddot{A}_M, \text{ with } G_n \in l_d(i, H) \\
 &\quad i(G_1) = i \\
 &\quad \forall G_{n+1} \exists A^C \in G_{n+1}: i(G_n) \in c(A^C), \text{ with } G_n, G_{n+1} \in l_d(i, H) \\
 &\quad (\forall A^C \in \ddot{A}_M \exists G_n \in l_d(i, H)) : (i(G_n) \in c(A^C) \Rightarrow A^C \in G_n)
 \end{aligned}$$

Similarly, we define the local upstream search as:

$$\begin{aligned}
 l_u: \ddot{D} \times \ddot{M} &\rightarrow P(\ddot{E}) \\
 l_u(i, H) &= \begin{cases} \emptyset, & \text{if } \nexists B \in \ddot{A}_M: t_e^A(B) = i \\ \{G_1, G_2, G_3, \dots\}, & \text{else, with:} \end{cases} \\
 &\quad t_e^A(B) = i(G_n) \Rightarrow B \in G_n, \forall B \in \ddot{A}_M, \text{ with } G_n \in l_u(i, H) \\
 &\quad i(G_1) = i \\
 &\quad G_{n+1} \ni \exists G_m \exists A^C \in G_m: i(G_{n+1}) \in c(A^C) \text{ with } G_{n+1}, G_m \in l_u(i, H) \\
 &\quad \bigvee_{G_n \in l_u(i, H)} (\exists B \in \ddot{A}_M, \exists A^C \in G_n: t_e^A(B) \in c(A^C)) \Rightarrow (\exists G_m \in l_u(i, H): B \in G_m)
 \end{aligned}$$

#### 6.4.2 Remote Search

A remote search occurs when a server asks another server to perform a local search on its database. The FoodFile interface exposing this feature offers bulk requests – requests for the atoms of multiple entity-IDs at once.

Remote searches are the core ingredient for the global search. It is complementary to a local search if the client provides the optional ID of a member which is likely to recognize the entity: In addition to the local search, the server handling the trace also requests a search from the specified remote instance.

The formal notation of the local search covers remote searches since the targeted member is provided as an argument.

#### 6.4.3 Necessity of Merging

From a procedural point of view, the global search is an iterative execution of remote searches, each of which returns an entity set. Due to information sharing and caching, the information atoms collected from remote searches may be redundant or represent multiple versions of the same information. To compute a consistent, nonredundant result, merging algorithms are required on both: entity and atom level. The reason for this is that, on an entity level, different members may know different aspects about the supply chain of a given good: Let  $i$  be an arbitrary entity identifier and  $H_1, H_2$  be two members.

$$\begin{aligned}
 K_1 &= \{B \in \ddot{A}_{H_1} : t_e^A(B) = i\} \\
 K_2 &= \{B \in \ddot{A}_{H_2} : t_e^A(B) = i\} \\
 \text{Then } K_1 &\text{ does not necessarily equal } K_2.
 \end{aligned}$$

On atom level, two atoms with same entity-id, atom-id and version carry the same atom information, but not necessarily the same signatures:

$$\neg \forall (B_1, B_2) \in \ddot{A}^2 \\ \left( (t_a^A(B_1), t_e^A(B_1), t_v^A(B_1)) = (t_a^A(B_2), t_e^A(B_2), t_v^A(B_2)) \Rightarrow t_s^A(B_1) = t_s^A(B_2) \right)$$

Merging of entity sets is decomposed into multiple levels of merging: atom merging, entity merging, entity set merging, respectively. Each merging mechanism builds on the prior one.

#### 6.4.4 Atom Merging

Let  $B_1, B_2$  be atoms with

$$(t_a^A(B_1), t_e^A(B_1), t_v^A(B_1)) = (t_a^A(B_2), t_e^A(B_2), t_v^A(B_2))$$

We define  $m^A$ , the merging function for atoms as

$$m^A: \ddot{A} \times \ddot{A} \rightarrow \ddot{A} \\ m^A(B_1, B_2) = (t_a^A(B_1), t_e^A(B_1), t_v^A(B_1), t_l^A(B_1), t_s^A(B_1) \cup t_s^A(B_2))$$

#### 6.4.5 Entity Merging

Let  $H_1, H_2$  be two members.

Let  $G_1 \in \ddot{E} : (B_1 \in G_1 \Rightarrow B_1 \in \ddot{A}_{H_1})$  and  $G_2 \in \ddot{E} : (B_2 \in G_2 \Rightarrow B_2 \in \ddot{A}_{H_2})$

Let  $i(G_1) = i(G_2)$

We define  $m^E$ , the merging algorithm for entities as

$$m^E: \ddot{E} \times \ddot{E} \rightarrow \ddot{E} \\ m^E(G_1, G_2) = \\ \quad \forall B_1 \in G_1 \text{ DO} \\ \quad \quad \text{IF } \exists B_2 \in G_2: t_a^A(B_1) = t_a^A(B_2) \wedge t_v^A(B_1) = t_v^A(B_2) \text{ THEN} \\ \quad \quad \quad G_2 \leftarrow (G_2 \setminus \{B_2\}) \cup \{m^A(B_1, B_2)\} \\ \quad \quad \text{ELSE} \\ \quad \quad \quad G_2 \leftarrow G_2 \cup \{B_1\} \\ \quad \quad \text{END-IF} \\ \quad \text{END-DO} \\ \text{RETURN } G_2$$

#### 6.4.6 Entity Set Merging & Flattening

Let  $H_1, H_2$  be two members.

Let  $J_1 \in P(\ddot{E}) : (B_1 \in G_1 \Rightarrow B_1 \in \ddot{A}_{H_1}, \forall G_1 \in J_1)$

and  $J_2 \in P(\ddot{E}) : (B_2 \in G_2 \Rightarrow B_2 \in \ddot{A}_{H_2}, \forall G_2 \in J_2)$

We define  $m^S$ , the merging algorithm for entities sets as

```

 $m^S: P(\ddot{E}) \times P(\ddot{E}) \rightarrow P(\ddot{E})$ 
 $m^S(J_1, J_2) =$ 
   $\forall G_1 \in J_1$  DO
    IF  $\exists G_2 \in J_2: i(G_1) = i(G_2)$  THEN
       $J_2 \leftarrow (J_2 \setminus \{G_2\}) \cup \{m^E(G_1, G_2)\}$ 
    ELSE
       $J_2 \leftarrow J_2 \cup \{J_1\}$ 
    END-IF
  END-DO
  RETURN  $J_2$ 

```

We define flatten, which flattens a set of entity sets to a single entity set:

```

 $flatten: P(P(\ddot{E})) \rightarrow P(\ddot{E})$ 
 $flatten(L) =$ 
  IF  $L = \emptyset$  THEN
    RETURN
  ELSE
     $let J \in L$ 
    RETURN  $m^S(J, flatten(L \setminus \{J\}))$ 
  END-IF

```

#### 6.4.7 Global Search

A client request for a trace triggers a global search. The first step of a global search is a local search and, if applicable, a remote search by the member optionally mentioned in the trace request.

The resulting atoms are collected, added to the knowledge base and the following information is extracted:

- Involved entities. As already described in 6.4.1, both, upstream and downstream traces lead to entities which require a follow up inquiry.
- Involved members. Member-IDs can be found in involvement atoms, signatures and the 'responsible' fields of creation, destruction and transfer atoms.

The global search then coordinates parallelized remote searches for all entities across all involved members. As the local search, this is a recursive process. It bottoms out when neither new entities nor new members are reported and added to the coordinator's knowledge base. For new entities, a local search is performed in addition to the remote searches. The global search algorithm ensures to never ask a remote instance for information it delivered already

during past iterations. This is achieved by keeping track of which member replied with which entities during previous queries.

We define  $\vec{K}$ , the universe of search directions containing the two elements  $u$  (representing upstream search) and  $d$  (representing downstream search):

$$\vec{K} = \{u, d\}$$

The following algorithm describes the global search leveraging definitions from the previous sections and 6.4.8.

```

search_global:  $\vec{K} \times P(\vec{E}) \times P(\vec{T}) \times P(\vec{D}) \rightarrow P(\vec{E})$ 
search_global( $k, last\_iteration\_result, query\_history, entity\_ids$ )
  tasks  $\leftarrow$  compute_tasks( $query\_history, entity\_ids$ )
   $\forall (H, U) \in tasks$  DO
     $\forall i_1, i_2, \dots \in U$  DO
       $result_H \leftarrow flatten(l_k(i_1, H), l_k(i_2, H), \dots)^{25}$ 
    END - DO
  END - DO
  result  $\leftarrow$  flatten( $result_{H_1}, result_{H_2}, \dots$ )
  IF result \  $last\_iteration\_result = \emptyset$  THEN
    RETURN  $last\_iteration\_result$ 
  END - IF
  query_history  $\leftarrow$  update_history( $query\_history, entity\_ids, \{result_{H_1}, result_{H_2}, \dots\}$ )
  query_history  $\leftarrow$  update_members( $result, query\_history$ )
  entity_ids  $\leftarrow$  update_ids( $k, result, entity\_ids$ )
  result  $\leftarrow m^S(result, last\_iteration\_result)$ 
  search_global( $K, result, query\_history, entity\_ids$ )

```

The following statement describes the initial call to trigger this recursive search algorithm. The term query denotes the placeholder for the entity-ID(s) to gather information about.

```
search_global( $k, \emptyset, \emptyset, \{query\}$ )
```

This breath-first search is designed to be tail-recursion-optimizable. The executor is then only required to maintain the information of the most recent search iteration – the call stack is not preserved. Additionally, performance is gained as information is never retrieved twice and circles are avoided.

#### 6.4.8 Functions & Types to support Global Search

A member activity is a tuple to capture one of the following (depending on usage context)

- which entities a member does not need to be queried (again)
- which entities a member should be queried about next

---

<sup>25</sup>Due to the chosen notation, the local search takes the member to query as the second parameter. Even though it is not explicitly mentioned, this step implicitly involves a call to  $f()$  to look up the URL of the respective member. See 6.3.

We define a member activity as a 2-tuple  $T: \ddot{M} \times P(\ddot{D})$ .

We define  $\ddot{T}$ , the universe of all member activities.

We define a function `compute_tasks` which takes a member activity set representing a query history and a list of entity-IDs being relevant for the current search to calculate which member needs to be queried about which id in the upcoming search iteration.

```

compute_tasks:  $\ddot{T} \times P(\ddot{D}) \rightarrow \ddot{T}$ 
compute_tasks(history, entity_ids) =
   $\forall (H, U) \in \text{history}$  DO
     $\text{next}_n \leftarrow \text{entity\_ids} \setminus U$ 
     $L_n \leftarrow (H, \text{next}_n)$ 
  END – DO
  RETURN history

```

Based on the search results received from each member, the history of that member and the list of relevant entity-IDs, we can compute a new member activity set representing what entities each member was queried about and replied with. It is important to unify the IDs queried for and the IDs of the response entities, because the search should not query this member again for either one.

```

update_history:  $\ddot{T} \times P(\ddot{D}) \times (\ddot{M} \rightarrow P(\ddot{E})) \rightarrow \ddot{T}$ 
update_history(history, entity_ids, member_results) =
   $\forall (H, U) \in \text{history}$  DO
     $\text{history} \leftarrow \text{history} \setminus \{(H, U)\}$ 
     $y \leftarrow \text{result}_H \in \text{member\_results}$ 
     $\forall G_1, G_2, \dots \in y$  DO
       $U \leftarrow (H, \{i(G_1), i(G_2), \dots\} \cup \text{entity\_ids})$ 
    END – DO
     $\text{history} \leftarrow \text{history} \cup \{(H, U)\}$ 
  END – DO
  RETURN history

```

In order to prepare for the next search iteration, the received entities need to be checked for new members which could potentially contribute to the process of information gathering. The `update_members` function takes an entity set (the current search result) and the member activity set to be enriched with additional members. It returns the updated member activity set.

```

update_members:  $P(\ddot{E}) \times \ddot{T} \rightarrow \ddot{T}$ 
update_members(result, query_history) =
  member_ids =  $\emptyset$ 
   $\forall G \in result$  DO
     $\forall B \in G$  DO
      IF  $t_I^A(B) = I_V$  THEN
        member_ids  $\leftarrow$  member_ids  $\cup I_V$ 
      ELSE – IF  $t_I^A(B) = I_C$ 
         $\forall t_I^A(B) = I_T \vee t_I^A(B) = I_R$  THEN
          member_ids  $\leftarrow$  member_ids  $\cup t_r^I(t_I^A(B))$ 
        END – IF
      END – DO
    END – DO
   $\forall m\_id \in member\_ids$  DO
    IF  $\nexists t_r^M(H) = m\_id, (H, \mathcal{U}) \in query\_history$  THEN
      query_history  $\leftarrow$  query_history  $\cup \{(m\_id, \emptyset)\}$ 
    END – IF
  END – DO
RETURN query_history

```

Each search iteration requires a list of relevant entity-IDs to query for. The function `update_ids` takes the search direction (upstream or downstream), the current search result and the current list of relevant IDs of which it returns the updated version. The chain direction determines which additional IDs are relevant for the upcoming search iteration: The IDs of the response entities themselves or the IDs of the entities consumed during the creation of the new search results.

```

update_ids:  $\ddot{K} \times P(\ddot{E}) \times P(\ddot{D}) \rightarrow P(\ddot{D})$ 
update_ids(dir, result, entity_ids) =
  IF dir = d THEN
    entity_ids  $\leftarrow$  entity_ids  $\cup (\forall_{G \in result} G \mapsto i(G))$ 
  ELSE
     $\forall G \in result$  DO
      IF  $\exists A^C \in G$  THEN
        entity_ids  $\leftarrow$  entity_ids  $\cup c(A^C)$ 
      END – IF
    END – DO
  END – IF
RETURN entity_ids

```

## 6.5 Labels & Supply Chain Linkage

Entities are tagged with their IDs using labels as discussed in 2.1.3. Sometimes, an entity-ID may not be sufficient to foster a trace. For example, imagine a jam factory ordering fruits from many suppliers. When a pallet of fruits arrives at the truck loading bay, their entity-ID is only useful if it is known which partner instance to query. FoodFile offers two options to improve supply chain linkage: If possible, the purchasing department should link entity-IDs of acquired



products to the supplier's member-ID as part of the ordering process leveraging the involvement atom type. This is not always practicable though. Food producers may therefore add their member-ID to the label. Upon execution of a trace, this is the parameter which can be passed in as the member which is likely to recognize the entity. As part of the receiving inspection and quality assurance, labels should be scanned and trigger an automatic creation of an involvement atom in order to persist the link between supplier and entity-ID.

As a third information in addition to entity-ID and member-ID, labels may also contain an access token. The purpose of this token is discussed in section 6.7.3. To increase authenticity, labels can be signed with the certificate provided by the membership service, if enough digital storage is available (e.g. on an RFID chip).

#### 6.5.1 Consumer Information

Producers of retail products may decide to label their goods with the respective entity-ID and their member-ID. This information enables every FoodFile instance to execute a global search without prior knowledge and without a local database. The necessary algorithm can be executed locally in a browser or as part of a smartphone app. One possible approach to apply the IDs to the product packaging are QR-Codes.

### 6.6 **Caching and Reconciliation**

A FoodFile server can be configured to cache information atoms from other servers. This can be all available atoms throughout the network for certain entities, or all information generated by a partner company. Caching may serve two purposes:

- Maintaining a local copy implies independency from others. Locally available data is not lost if other members encounter technical problems or file bankruptcy. Redundancy is, among signatures and reconciliation, the driving factor for data integrity in a FoodFile network.
- Local persistence of partner information leads to a shortened response time during a trace because less requests need to be made. This improves the user experience, especially in the context of consumer information (6.5.1).

When multiple parties are involved in a process, the same information may be generated by those parties independently. The shipping of goods, for example, may be logged by the sender, the receiver and the carrier while the generated atoms can differ: The sender and receiver may assume the coordinates of the facilities as start- and endpoints while the carrier probably relies on GPS. When a contradiction occurs, FoodFile tries to resolve the conflict by aligning the different values using averages or medians. This automated process remains within the boundaries chosen and configured by the hosters of the involved servers.

If a conflict cannot be solved automatically, it gets reported to the conflict dashboards of the involved participants in order to be handled manually.

The adjusted information atom receives the timestamp of adjustment as value for the version field. It is inevitably higher than all timestamps of the previous conflicting versions. Whenever multiple parties agree on a stored fact, they should sign it.

The process described does not enforce eventual consistency throughout the network because of two reasons:

- A digital system cannot legitimately decide between two conflicting opinions about the truth.
- Reconciliation only happens when contradictions are detected, for example during cashing or if both parties have an active interest in consistent information<sup>26</sup>.

Consequently, a trace may sometimes yield conflicting atoms. In those cases, it remains the decision of the respective client on how to visualize this. Some options are:

- Rely on the company's own signature, if applicable
- Configure automated reconciliation rules to be applied on the knowledge base before visualization
- Rely on the atom with the most signatures
- Rely on the atom with the highest version number
- Rely on the atom with a signature from a trusted partner
- Visualize the conflict

It is important to note that preferring an information atom based on its version field's value is only reasonable when one trusts the signing participants.

## 6.7 Sharing and Security

While access restrictions are enforced on atom level, they can be configured in a rule-based manner. For example, a company may decide to make all product descriptions (atoms of type description) publicly available but share additional information with specific partners only. In that scenario, FoodFile decides which atoms to deliver based on which member submitted the request. Information atoms can be shared publicly, not at all, with selected partners, public authorities or based on whether the requestor was involved in a supply chain process targeting the entity which is subject of the request.

### 6.7.1 Sharing with partners or public authorities

Sharing configurations based on partners or authorities are enforced by signed requests. The member issuing a request signs it as it would sign an atom with its X.509 private key. Certifi-

---

<sup>26</sup> It can be considered to introduce a mechanism, enabled by default, which continuously synchronizes atoms with multiple stakeholders.

cate validity can be verified through the membership service. As a result, the requested FoodFile instance has certainty about the identity of the requestor and may decide what atoms to deliver based on the configured rules.

#### 6.7.2 Sharing with suppliers

Sharing data with an upstream supplier of an entity is a continuation of this principle. Having certain knowledge about the identity of a given request, FoodFile needs to validate that the requestor was indeed a supplier of the product or ingredient s/he requests information about. This can be inferred by searching for the corresponding creation atoms. The mechanism requires that a creation atom based on which a sharing decision is made is related to an involvement atom signed by the member handling the request<sup>27</sup>. Consequently, this sharing rule can only be effective at companies having at least one supply chain linkage mechanism in place (see 6.5).

#### 6.7.3 Sharing with buyers

While sharing data with downstream receivers of an entity could be handled analog to what has been described previously, as mentioned in 6.5, it is also possible to include an access token on the label attached to the physical product. In those cases, requests contain the access token from the label. The receiving FoodFile instance recognizes the token it initially issued upon generation of the label and delivers the atoms the token owner is privileged to see. Tokens can be configured to expire after a given time interval.

#### 6.7.4 Re-Sharing

Once an information atom is shared, it can be cached by the receiving party. If the re-sharing flag is set, after being cached, the information atom will be governed by the sharing policy of the receiving party. If the re-sharing flag is not set, the FoodFile instance of the receiving party will keep the information private.

It is important to note that re-sharing forbiddance cannot be guaranteed as it is a FoodFile feature and not an implication of the imposed security.

#### 6.7.5 Attacks

Malicious participants may decide to emit atoms containing wrong information. Consequently, a FoodFile client may decide not to visualize unsigned atoms and servers may be configured to only perform follow-up searches based on signed atoms.

Wrong information atoms are specifically problematic when they are designed to attack a FoodFile instance carrying out a trace. During an upstream trace, for example, an attacker may

---

<sup>27</sup> This restriction prevents the following attack: An arbitrary member signs a fictional creation atom mentioning the member itself in the responsible field. If FoodFile decided to share data based on that atom, this would be tantamount to sharing the data with everyone, as everyone could pretend to be the supplier.

design creation atoms leading to a large number of inbound entities and hence creating a snowball effect of requests. In addition to only perform algorithmic decisions based on signed information, the global search has a threshold of iterations and involved participants requiring a manual approval for continuation when passing it. A trustworthy membership provider with a reasonable accreditation process remains a key factor in attack prevention.

## **6.8 Client**

FoodFile has an interface which allows to connect web-based clients, apps, business intelligence tools and other third-party software. The interface consumes requests for local or global searches and delivers the resulting information atoms. A FoodFile client in the narrower sense should be able to scan labels, trigger searches and visualize the resulting atoms appropriately, for example as an ingredient tree enriched with shipping information.

## 7 Comparing Concept, Formal Model and Implementation

Chapter 6 describes the architectural concept behind FoodFile. The major ideas are backed by a formal model composed of datatypes and algorithms which is introduced alongside with the textual description. The provided MVP implementation of the described concept is superimposable with the formal model. The implementations of those aspects not covered by the formal notation deviate in certain aspects from the concept. Table 3 compares concept, formal model and implementation based on the requirements from 5.1. It provides further information and reasoning on differences.

Objective	Model	Concept	Implementation
Whole chain traceability for individual entities	Superimposable, see section 7.1		
Bidirectional tracing			
On-Site data storage, if desired			
Information sharing regulation for individual pieces of information regarding specific entities	The formal model satisfies the core objectives. It serves as a foundation but does not cover the additional concepts around privacy and trust.	See 7.2 for an explanation of the deviations.	
Information accessibility for non-participants (e.g. consumers)			
Replication to the degree desired by the participants		See 7.3 for an explanation of the deviations.	
Contradicting Information is detectable		See 7.4 for an explanation of the deviations.	
Resilient histography of data changes			

Table 3: comparing model, concept and implementation

### 7.1 Core Objectives and on-site data storage

Model, concept and implementation are superimposable regarding the core objectives and on-site data storage with one exception: The model assumes the existence of a single membership service to simplify notation. The concept describes the possibility of the existence of

multiple membership services and the implementation allows to configure which membership service should be connected to.

The implementation supports all information atom types and offers all described APIs for data supply, instance communication and client communication. Up- and downstream tracing functionality is available as described, including local-, remote- and global search, automatic recursion and atom merging. Supply chain linkage (6.5) and consumer information (6.5.1) is supported in the sense that the server can conduct traces without a local database, if the request contains the optional member-ID. The client can submit requests for up- and downstream traces and assemble the resulting atoms to a tree structure of transformations and transportations. The visualization adjusts dynamically to the available information, e.g. a missing description for an entity falls back to its ID, etc. The membership service mode offers an API to register and look up members, the client comprises a web form to sign up.

## 7.2 Information sharing regulation

The concept describes a rule-based sharing approach involving X.509 certificates to prove identities and dedicated access validation strategies for sharing with partners, public authorities, suppliers and buyers.

The implementation is limited to the solution presented in 6.7.3. Atoms comprise a flag which distinguishes between 4 sharing modes: enabled, disabled, by-token, by-token-or-chain. When sharing is set to enabled, the atom will be shared with other members during remote searches and caching requests. When sharing is set to disabled, the FoodFile instance will never expose the atom through the public API. When sharing is set to by-token, the atom will be shared only if the requester presents a token for the entity the atom belongs to. If the sharing is set to by-token-or-chain, the atom in question is shared if either case is true:

- The requester presents a token for the entity the atom belongs to
- The atom is part of a query result and the requester presented a token for the entity the query was about.

### 7.2.1 Tokens

The token for an entity is a SHA-256 hash of the concatenation of entity-ID and a member specific secret. The secret is provided as a start-up parameter. With this approach, there is no need to store tokens for entities as they can always be computed on-the-fly. A member cannot guess the token as this would require the knowledge of the secret. The secret cannot be computed given the entity-ID and the token due to the characteristics of secure hash functions such as SHA-256.

### 7.2.2 Instance level user management

A user management on instance level is not included in the conceptual work of this paper and not part of the implementation. For a practical use case, it will be required. As a result of a missing login, the private API, which should be exposed to the FoodFile client of the respective member only, is not protected. A side effect is that any sharing regulation mechanism cannot be effective from a security standpoint.

## 7.3 Data Replication

Section 6.7 describes automatic information retrieval from a configurable set of partners with a combination of automated and manual conflict resolving strategies.

The implementation offers a possibility to manually trigger a process which copies the current state of one or multiple other FoodFile instance(s) (sharing regulations apply) into the own database. It does not offer any reconciliation. Instead, in case of a conflict, the entity merging algorithm (6.4.5) will resolve it in favor of either one of the two conflicting atoms.

## 7.4 Histogrammy resilience and conflict detection

As the model is designed to never delete information (see 6.2.6), histogrammy is an implicit result of auditing (see 6.2, version field). Visualizing the histogrammy is taken care of by the FoodFile client (the model and the implementation are designed to ship all retrieved information for an entity, including historical data).

Resilience correlates with the amount of copies available for the information as a result of data replication: The more copies available, the more obvious is a single deviation. Given that copies of an atom (equal atom-id, entity-id and version) are available, resilience is a matter of detecting contradictions.

If historic versions for an atom exist, the FoodFile client implementation allows the user to interactively select which version should be used for the visualization. The tree-chart adjusts automatically. Atoms (atom versions) can be verified by clicking a verify button. FoodFile then queries all members involved in the specific search for their hashes of the atom (version) in question. The client presents the user with statistics about the atom (version) unveiling any deviations regarding this atom (version) among the involved members. See 8.4.13 - 8.4.17 for examples.

## 7.5 Implementation Details

The reference implementation is built on ASP.NET Core 3.1 MVC and written in F#. Elasticsearch serves as the database. The Client is a React Single Page Application (SPA) with Boot-

strap and D3 for graph visualizations. Communication among FoodFile instances and communication between server and client or data supplying systems happens through RESTful APIs. The system ships as a Docker image. There is one unified implementation which can be started as regular FoodFile instance, as a membership service or in a combined mode for development purposes. On the client side, the SPA's menu adjusts accordingly.

Information atoms are modeled as JSON objects, entities are represented by lists of information atoms belonging to the same entity-ID. Database documents are entities: A database documents represents the entire knowledge (all information atoms) the respective FoodFile instance has about a given entity.

Elasticsearch is the database of choice because it has scalability capabilities, works with JSON documents and, due to the underlying Lucene library, can be expected to deliver good compression results. Lucene is not optimal for frequent delete operations. This drawback does not apply for the given use-case as there is no delete. [74] [75] [76]

F# was chosen as the programming language because it combines the functional paradigm with the interoperability with all libraries of .NET Core. It runs on all major platforms and has a large developer community. F#, as most other functional languages, offers strong convenience features such as pattern matching and is less error prone because of its explicit null value handling and sophisticated compile time checks. [77] [78]

React is the leading framework for web-based frontends. As HTTP requests are controlled by client-side JavaScript once the SPA is downloaded, React works well with ajax technologies. One future use-case is to deliver real time updates of the current trace, e.g. using Websockets, and constantly re-render the result-visualization. [79] [80]



## 8 Evaluation

The proposed concept is evaluated by conducting test scenarios against the provided implementation. The scenarios are carried out in a simulated test context consisting of one membership service, three members and one non-member instance. The non-member instance is relevant to show that information can be made accessible for 3<sup>rd</sup> parties such as consumers. The instances are deployed independently and communicate exclusively through their FoodFile APIs. A test scenario consists of

- Purpose: a reference to the objective(s) (see section 5.1) the scenario validates
- Setup: the individual knowledge bases the respective members should hold at the beginning of the scenario simulation.
- Scenario library: the name of the used knowledge base distribution in the scenario library implementation.
- Trigger: one or multiple interactions with the system (i.e. with the instances)
- Acceptance criteria: Desired outcomes, for example, manifest in correct visualizations of traces which span across multiple members.

Section 8.1 introduces a tool to automate step two and make it error prone. Section 8.2 covers the evaluation implementation and deployment details. Sections 8.3 and 8.4 specify the exemplary supply chain, test atoms, test scenarios, their results and a reviews against the acceptance criteria.

### 8.1 Knowledge base distribution

All atoms required for the evaluation are specified in a single file, the atom library. This library can store contradictive information, e.g. a copy of an atom where the sharing policy is configured differently.

The atom library is the foundation for the scenario library. In terms of the knowledge distribution tool, a scenario consists of multiple knowledge bases to be injected into the respective instance databases. A knowledge base is a list of references to atoms in the atom library. Scenarios can be re-used to build other, more advanced scenarios, see Figure 2 for an example. For successful execution, the tool needs to be configured with the name of the desired scenario and the URLs to the instance databases in which the knowledge bases should be injected. First, it removes all information currently stored in the instance databases. Then it re-populates with the new information.

### 8.2 Implementation & Deployment

As FoodFile, the distribution tool is written in F#. The atoms, of which examples are provided in Figure 1 are in fact atom templates: Being implemented as a function, they take the IDs of

the respective FoodFile members as a parameter. This approach keeps the evaluation portable since these IDs can change upon each deployment of the simulation context<sup>28</sup>. Member-IDs are relevant for the involvement atom type and atom types with `responsible`-fields.

The distribution tool leverages the FoodFile source code as a library to assemble atoms to entities and for communication with Elasticsearch, the database technology FoodFile is built on.

Figure 1 shows one creation atom template (top) and one transfer atom template (bottom). The Creation template describes the production of preserving sugar mentioning two inbound entities (strawberries and gallant), whereas `sb_1` and `ge_1` are placeholder variables for the entity-IDs of these ingredients. The transfer template describes the transport of the preserving sugar to the jam production facility, which is the only track point in the list. Analogue to the creation, the transfer features a `responsible`-field, populated with the arguments of this template function.

```
let ps_1_cr_1 fffId ssId yjId = {
  AtomID = "BCWG"; EntityID=ps_1; Version=1;
  Information = Creation ({
    InEntities = [sb_1; ge_1];
    Location = None
    Responsible = Some(ssId);
    Timestamp = 1555303285L});
  Signatures= [];
  Sharing=Enabled}

let ps_1_tr_1 fffId ssId yjId = {
  AtomID = "NCDQ"; EntityID=ps_1; Version=1;
  Information = Transfer ({
    Responsible=Some(ssId);
    TrackPoints= [
      ({ Name=Some("YummyJam Facility");
        Coordinates="51.590067, 8.1050100"},
        1555598311L)]
    });
  Signatures= [];
  Sharing=Enabled}
```

Figure 1: atom library excerpt

Figure 2 shows two example scenarios called `basic` and `basic_linked`. The `basic` scenario assigns each of the three members in the test context a basic knowledge base about a very simple jam production example: The `FreshFruitFarmers` member holds three pieces of information: The harvesting of a particular strawberry pallet, the transfer to the jam production facility and a textual description. The `SugarSilo` member has information about a pallet of sugar beet (creation and description), a unit of gallant (description only) and the outbound

<sup>28</sup> Member-IDs cannot be chosen manually but are assigned by the membership service when a member is registered.

entity: preserving sugar (creation out of sugar beet and gallant, transfer to the jam production facility, description). The YummyJam member has the creation and description of five jam jars assigned. Chapter 8.3 provides an introduction in the exemplary supply chain used for this evaluation. The `CapturePrepare` parameter is for additional atoms to be printed to the console when executing the tool. The output can be used to test the FoodFile Capture API (8.4.4).

The `basic_linked` scenario is built on top of the basic scenario and adds involvement atoms to the knowledge base of each member, making sure the instances know who to reach out to during a trace. Most testing scenarios described in 8.4 are built on top of either of these two.

```
let basic = {
  FreshFruitFarmers = [st_1_cr_1; st_1_tr_2; st_1_ds_1]
  SugarSilo =          [sb_1_cr_1; sb_1_ds_1]           // Sugar bett
                      @ [ps_1_cr_1; ps_1_tr_1; ps_1_ds_1] // Preserving Sugar
                      @ [ge_1_ds_1]                   // Gellant
  YummyJam =          [jj_1_cr_1; jj_1_ds_1]
                      @ [jj_2_cr_1; jj_2_ds_1]
                      @ [jj_3_cr_1; jj_3_ds_1]
                      @ [jj_4_cr_1; jj_4_ds_1]
                      @ [jj_5_cr_1; jj_5_ds_1]
  CapturePrepare =    []
}

let basic_linked = {
  FreshFruitFarmers = basic.FreshFruitFarmers
                      @ [ff_1_iv_1] // Strawberries
  SugarSilo =        basic.SugarSilo
                      @ [ss_1_iv_1] // Preserving Sugar
  YummyJam =         basic.YummyJam
                      @ [jj_1_iv_1] // Preserving Sugar
                      @ [jj_2_iv_1] // Strawberries
  CapturePrepare =    []
}
```

Figure 2: scenario library excerpt

The test simulation was carried out on a Kubernetes cluster hosting three deployments (no replicas), one per member. Each pod contains the FoodFile container<sup>29</sup> and one Elasticsearch container<sup>30</sup>. The deployment specification is available as part of this work<sup>31</sup>. FoodFile.org served as membership service, hosted separately on a virtual machine. The evaluation can be

<sup>29</sup> Dockerfile is available as part of this work. Alternatively refer to Docker Hub: leonardkoll/foodfile:thesis

<sup>30</sup> Available on Docker Hub: elasticsearch:7.5.0

<sup>31</sup> See FoodFile/Evaluation/deployment-supplychain.yml

reproduced with a custom membership service by starting FoodFile in membership mode as an additional deployment<sup>32</sup>.

### 8.3 Exemplary Supply Chain

The supply chain used for this evaluation is devised around strawberry jam production. In the most simplistic case, strawberry jam is made of strawberries and preserving sugar, the preserving sugar is made of sugar and gellant. In the example, a pallet of strawberries gets harvested and shipped to the jam production facility by the FreshFruitFarmers company<sup>33</sup>. The preserving sugar is produced by SugarSilo, which also oversees the harvesting of the sugar beet. SugarSilo buys the required gellant from a company which does not participate in the FoodFile network. SugarSilo decides to register the gellant as an ingredient but does not store any information except the description 'Gellant'. YummyJam is a producer of jam (jam jars) and creates five jam jars from the strawberry pallet and the preserving sugar. Table 4 provides an overview of all physical entities involved in the process and their assigned IDs which are consistent across all scenarios (and screenshots) in this work.

Physical entity	Entity-ID
Strawberry pallet	8EALKMS2Q7
Sugar beet pallet	UGSESG22LL
Preserving sugar	EIDDTPPDB2
Gellant	KVDBFFSAB4
Jam jars 1-5	6FGEMO4WX8, PMK9BG1YL5, JVE8F98148, LL9ZOFC5EH, 1I2VHCWKJL

Table 4: entity-IDs

Figure 3 shows a tree visualization of the described process. The tree references a specific jam jar at the root level, but the process is equal for all five jam jars (8.4.3 contains a visualization). This section is an introduction to the chosen evaluation context. The concrete supply chain is altered in the respective scenarios.

<sup>32</sup> See FoodFile/Evaluation/deployment-membership.yml

<sup>33</sup> All company names are fictional.

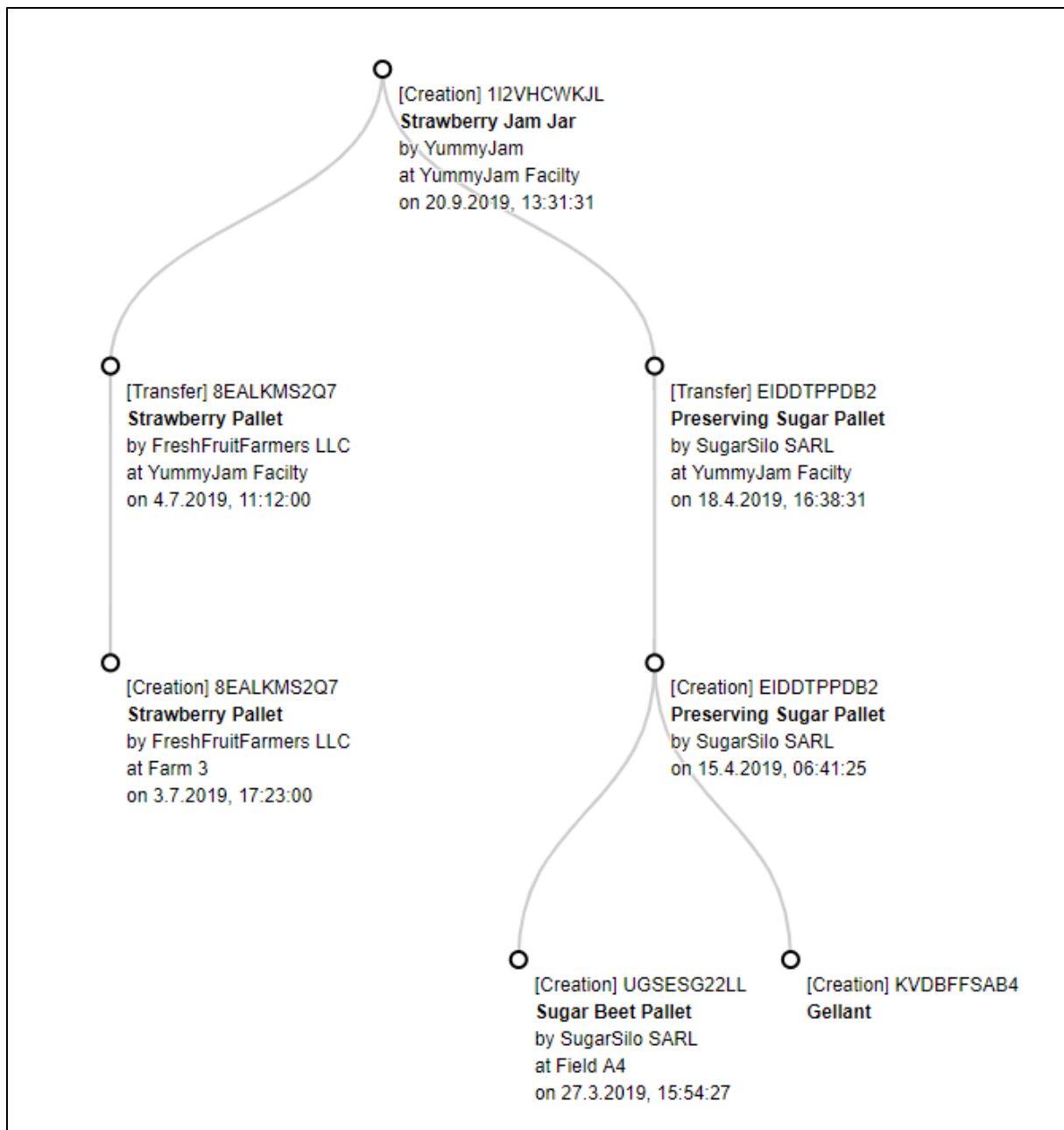


Figure 3: exemplary supply chain

### 8.3.1 Atom descriptions

The atom library contains a total of 37 atom templates which, as a groundwork for section 8.4, are listed in Table 5. If not specified otherwise, atoms have sharing set to enabled and version set to 1.

Atom template name <sup>34</sup>	Type	Entity	Description
st_1_cr_1	Creation	Strawberry pallet	Harvest, no inbound entities.
sb_1_cr_1		Sugar beet pallet	Harvest, no inbound entities.
sb_1_cr_2			As sb_1_cr_1 with modified location.
sb_1_cr_3			As sb_1_cr_1 with sharing policy set to disabled.
ps_1_cr_1		Preserving sugar	Inbound entities: sugar beet, gellant. No location provided.
jj_1_cr_1		Jam jar 1	Inbound entities: strawberry pallet, preserving sugar.
jj_1_cr_2			As jj_1_cr_1 with no inbound entities.
jj_1_cr_3			As jj_1_cr_1 with version field set to 2.
jj_2_cr_1		Jam jar 2	As jj_1_cr_1.
jj_3_cr_1		Jam jar 3	
jj_4_cr_1		Jam jar 4	
jj_5_cr_1		Jam jar 5	
ge_1_cr_1		Gellant	No inbound entities. No responsible member.
ge_1_cr_2			As ge_1_cr_1 but with version set to 2 and with responsible field set to an invalid member-ID.
ge_1_cr_3			As ge_1_cr_1 but with an increased timestamp.
ge_1_cr_4			As ge_1_cr_2 but with an increased timestamp.
ge_1_cr_5			As ge_1_cr_4 but with version set to 2.
st_1_tr_1	Transfer	Strawberry pallet	Transport to the FreshFruitFarmers warehouse
st_1_tr_2			Transport to the YummyJam facility
st_1_tr_3			As st_1_tr_1 with sharing policy set to by-token.
st_1_tr_4			As st_1_tr_2 with sharing set to disabled.
ps_1_tr_1		Preserving sugar	Transport to the YummyJam facility
st_1_ds_1	Description	Strawberry pallet	Regular description atom.
st_1_ds_2			As st_1_ds_1 with sharing set to disabled.
sb_1_ds_1		Sugar beet	Regular description atom.
ge_1_ds_1		Gellant	
ps_1_ds_1		Preserving sugar	
jj_1_ds_1		Jam jar 1	
jj_2_ds_1		Jam jar 2	
jj_3_ds_1		Jam jar 3	
jj_4_ds_1		Jam jar 4	
jj_5_ds_1		Jam jar 5	
ff_1_iv_1	Involvement <sup>35</sup>	Strawberry pallet	YummyJam
ss_1_iv_1		Preserving sugar	YummyJam
jj_1_iv_1			SugarSilo
jj_2_iv_1		Strawberry pallet	FreshFruitFarmers
jj_2_iv_2			Invalid member-ID

Table 5: atom library

<sup>34</sup> The name is referenced from the scenario library. It must not be confused with the atom identifier: The atom library may contain multiple examples of the same atom ( $\triangleq$  same identifier) with some modifications. While the combination of entity-ID, atom-ID and atom version is unique in the FoodFile model, it is not necessarily unique in the atom library as it contains atom templates for multiple test scenarios.

<sup>35</sup> The description column only mentions the involved company. It reads as: ‚Mentions company X to have been involved in production or processing of this physical entity‘

## 8.4 Scenarios

Each objective is tested by one or multiple scenarios which aim to cover all of its implications. Table 6 shows the relationship between objectives and scenarios.

Objective	Scenarios
Whole chain traceability for individual entities	8.4.2, 8.4.3, 8.4.4, 8.4.5, 8.4.7, 8.4.8
Bidirectional tracing	
On-Site data storage, if desired	8.4.1
Information sharing regulation for individual pieces of information regarding specific entities	8.4.10, 8.4.11
Information accessibility for non-participants (e.g. consumers)	8.4.9
Replication to the degree desired by the participants	8.4.6
Contradicting Information is detectable	8.4.12, 8.4.13
Resilient historiography of data changes	8.4.14, 8.4.15, 8.4.16, 8.4.17

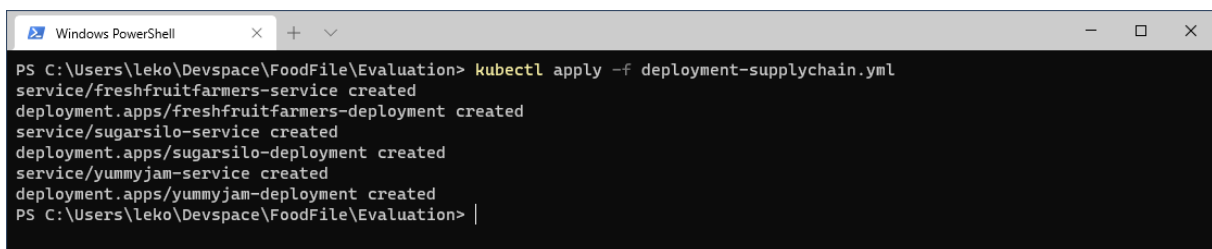
Table 6: scenario overview

### 8.4.1 Deployment

**Purpose:** By being able to deploy FoodFile for three individual members with their respective databases each, it is shown that the developed solution does not require a centralized data store. If desired, it allows for on-site data storage and data processing, since the database and the FoodFile server can be hosted on premises for each member individually.

**Setup:** The deployment deviates from other scenarios in that it does not involve any knowledge base injection. Required are only the Kubernetes deployment specifications which are available as part of this work. If one does not want to rely on the FoodFile image from Docker Hub, the repository and the Dockerfile available in the project folder can be used to build and register the project in any container registry.

**Trigger:** The deployment is triggered with `kubectl apply` (Figure 4). No further action is required as Elasticsearch and FoodFile start automatically upon container creation.



```

PS C:\Users\leko\Devspace\FoodFile\Evaluation> kubectl apply -f deployment-supplychain.yml
service/freshfruitfarmers-service created
deployment.apps/freshfruitfarmers-deployment created
service/sugarsilo-service created
deployment.apps/sugarsilo-deployment created
service/yummyjam-service created
deployment.apps/yummyjam-deployment created
PS C:\Users\leko\Devspace\FoodFile\Evaluation>

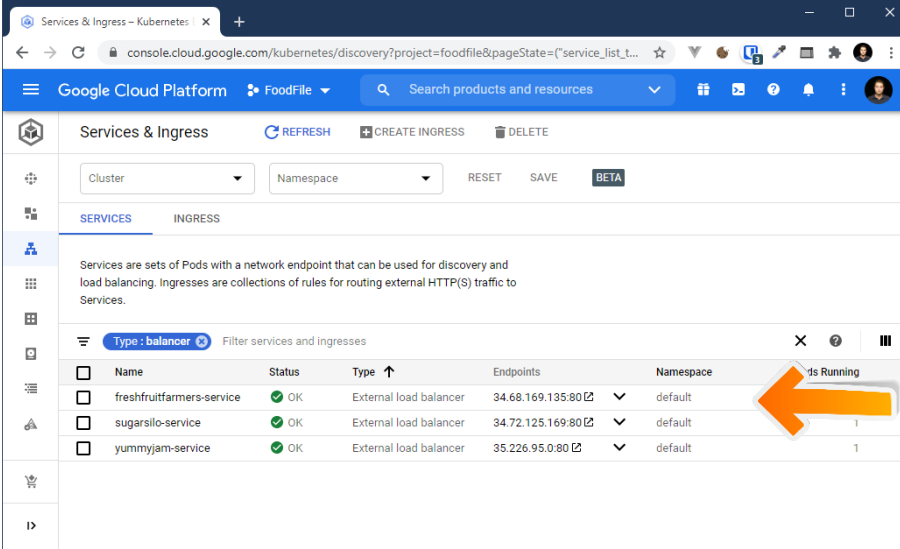
```

Figure 4: deploying the evaluation with Kubernetes

**Acceptance criteria:**

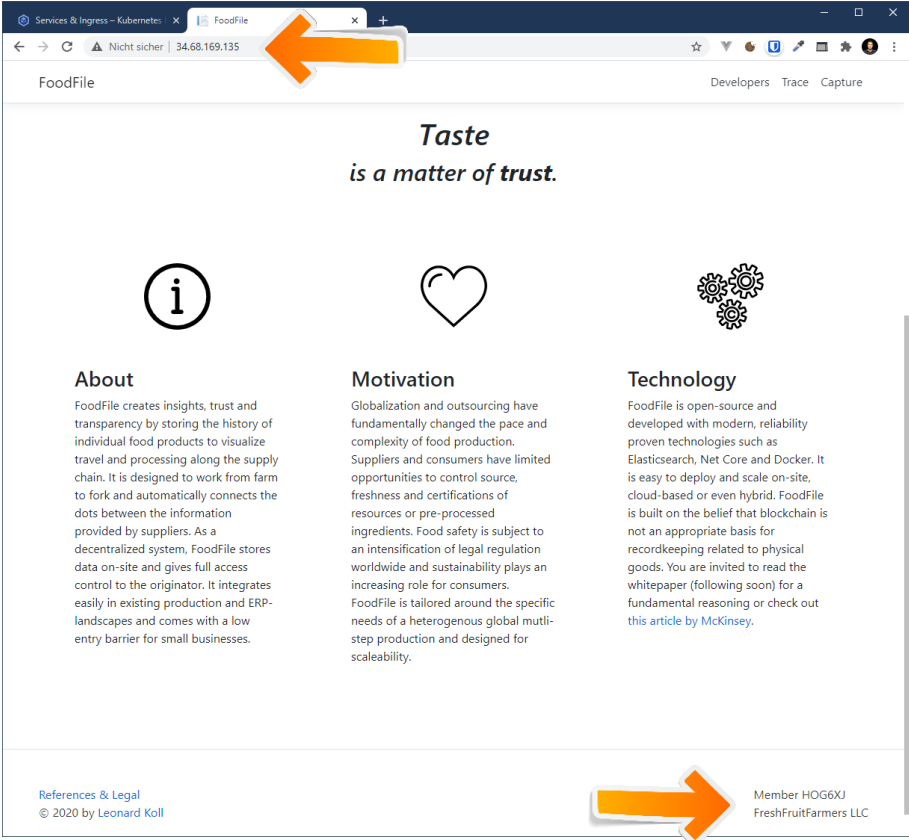
- FoodFile instances can be started in logically separated environments.
- Each instance has its own database as opposed to a shared one.
- FoodFile can be made accessible through public URLs.

**Results:** After executing `kubectl apply`, the Kubernetes load-balancer services become available and the member instances web interfaces are accessible through on specified IP addresses (Figure 5, Figure 6).



Name	Status	Type	Endpoints	Namespace	Pods Running
freshfruitfarmers-service	OK	External load balancer	34.68.169.135:80	default	1
sugarsilo-service	OK	External load balancer	34.72.125.169:80	default	1
yummyjam-service	OK	External load balancer	35.226.95.0:80	default	1

Figure 5: deployment management



**Taste is a matter of trust.**

**About**

FoodFile creates insights, trust and transparency by storing the history of individual food products to visualize travel and processing along the supply chain. It is designed to work from farm to fork and automatically connects the dots between the information provided by suppliers. As a decentralized system, FoodFile stores data on-site and gives full access control to the originator. It integrates easily in existing production and ERP-landscapes and comes with a low entry barrier for small businesses.

**Motivation**

Globalization and outsourcing have fundamentally changed the pace and complexity of food production. Suppliers and consumers have limited opportunities to control source, freshness and certifications of resources or pre-processed ingredients. Food safety is subject to an intensification of legal regulation worldwide and sustainability plays an increasing role for consumers. FoodFile is tailored around the specific needs of a heterogenous global multi-step production and designed for scalability.

**Technology**

FoodFile is open-source and developed with modern, reliability proven technologies such as Elasticsearch, Net Core and Docker. It is easy to deploy and scale on-site, cloud-based or even hybrid. FoodFile is built on the belief that blockchain is not an appropriate basis for recordkeeping related to physical goods. You are invited to read the whitepaper (following soon) for a fundamental reasoning or check out [this article by McKinsey](#).

References & Legal  
© 2020 by Leonard Koll

Member HOG6XJ  
FreshFruitFarmers LLC

Figure 6: deployment accessibility



To ease navigation and readability, DNS is configured for each member instance declared in Table 7. These URLs are visible across all browser screenshots.

Test scenario member	URL
FreshFruitFarmers	foodfile.freshfruitfarmers.com
SugarSilo	sugarsilo.evorne.io
YummyJam	yummyjam.evorne.io

Table 7: test scenario member URLs

It can be verified that the deployment of each member contains its own database. Figure 7 does this exemplary for FreshFruitFarmers.

The screenshot displays the 'Pod details' page for a deployment named 'freshfruitfarmers-deployment-564ddf455-n7tbs' in the 'default' namespace. The deployment is in the 'Running' phase. The 'Containers' section lists two containers: 'elasticsearch' and 'foodfile', both running. An orange arrow points to the 'foodfile' container, highlighting it as the database component. The 'Exposing services' section shows a 'freshfruitfarmers-service' of type 'Load balancer'.

Figure 7: individual databases per member

### 8.4.2 Tracing Abilities

**Purpose:** The scenario should verify

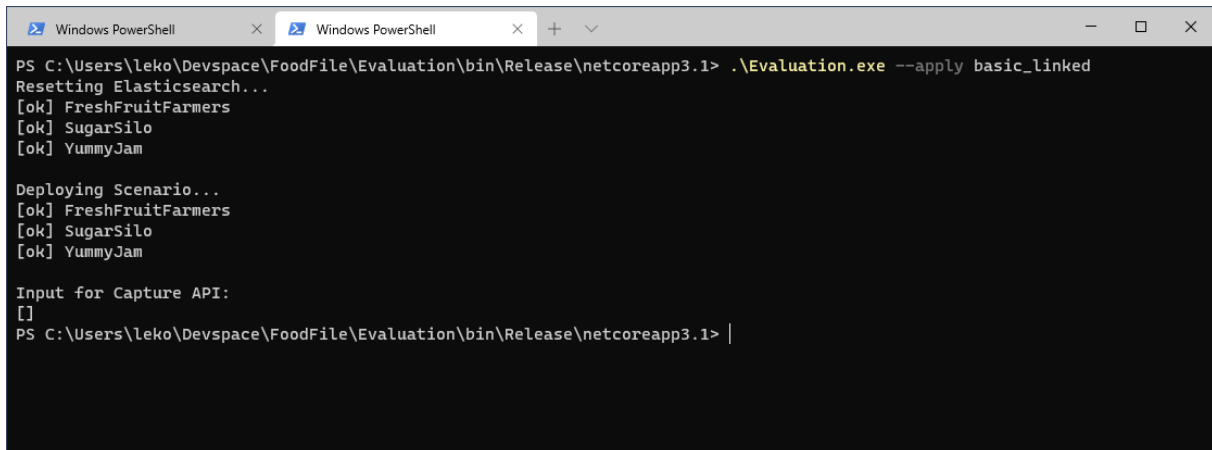
- the ability to trace a consumer facing good to its production origins (upstream) and from any of these origins back to all resulting entities (downstream).
- That a trace can successfully be coordinated across multiple supply chain members.

**Setup:** the knowledge base distribution of this scenario is designed around the assumption that each supply chain member stores information about its own value creation steps. Involvement atoms are added to the knowledge bases for the respective direct upstream and subsequent supply chain members. Table 8 shows the distribution of atoms to members.

Member	Injected knowledge base
FreshFruitFarmers	st_1_cr_1; st_1_tr_2; st_1_ds_1; ff_1_iv_1
SugarSilo	sb_1_cr_1; sb_1_ds_1; ps_1_cr_1; ps_1_tr_1; ps_1_ds_1; ge_1_ds_1; ss_1_iv_1
YummyJam	jj_1_cr_1; jj_1_ds_1; jj_2_cr_1; jj_2_ds_1; jj_3_cr_1; jj_3_ds_1; jj_4_cr_1; jj_4_ds_1; jj_5_cr_1; jj_5_ds_1; jj_1_iv_1; jj_2_iv_1

Table 8: atom distribution of the 'Basic Abilities' scenario

The scenario is applied leveraging the knowledge distribution tool. As this step is equal for all scenarios following, it won't be mentioned again. Figure 8 shows exemplary how the knowledge distribution tool is used.



```

PS C:\Users\leko\Devspace\FoodFile\Evaluation\bin\Release\netcoreapp3.1> .\Evaluation.exe --apply basic_linked
Resetting Elasticsearch...
[ok] FreshFruitFarmers
[ok] SugarSilo
[ok] YummyJam

Deploying Scenario...
[ok] FreshFruitFarmers
[ok] SugarSilo
[ok] YummyJam

Input for Capture API:
[]
PS C:\Users\leko\Devspace\FoodFile\Evaluation\bin\Release\netcoreapp3.1> |

```

Figure 8: knowledge distribution tool

**Scenario library:** basic\_linked

**Trigger:** The upstream trace for a jam jar is triggered by navigating to the FoodFile instance of YummyJam, selecting the Trace option from the menu, entering an ID of one of the jam jars (e.g. 1I2VHCWKJL) and opting for 'trace to origin'.

If the upstream trace results are as expected, the following downstream trace is triggered by arbitrarily choosing one of the origins (either strawberries, sugar beet or gellant), navigating

to the instance of the responsible member and executing a ‘trace to shelf’ for the respective entity-ID<sup>36</sup>.

#### Acceptance criteria:

- The upstream trace must visualize all entities involved in the production of the chosen jam jar, all involved supply chain members and all information (production, description and transfers) collectively known by the example participants.
- The downstream trace must visualize the same information as the upstream trace regarding entities of which the production process directly or indirectly consumed the specified entity.

#### Results:

Figure 9 shows the upstream trace result for entity 1I2VHCWKJL (jam jar 5). Figure 10 shows the downstream trace result for entity UGSESG22LL (sugar beet). The acceptance criteria are met.

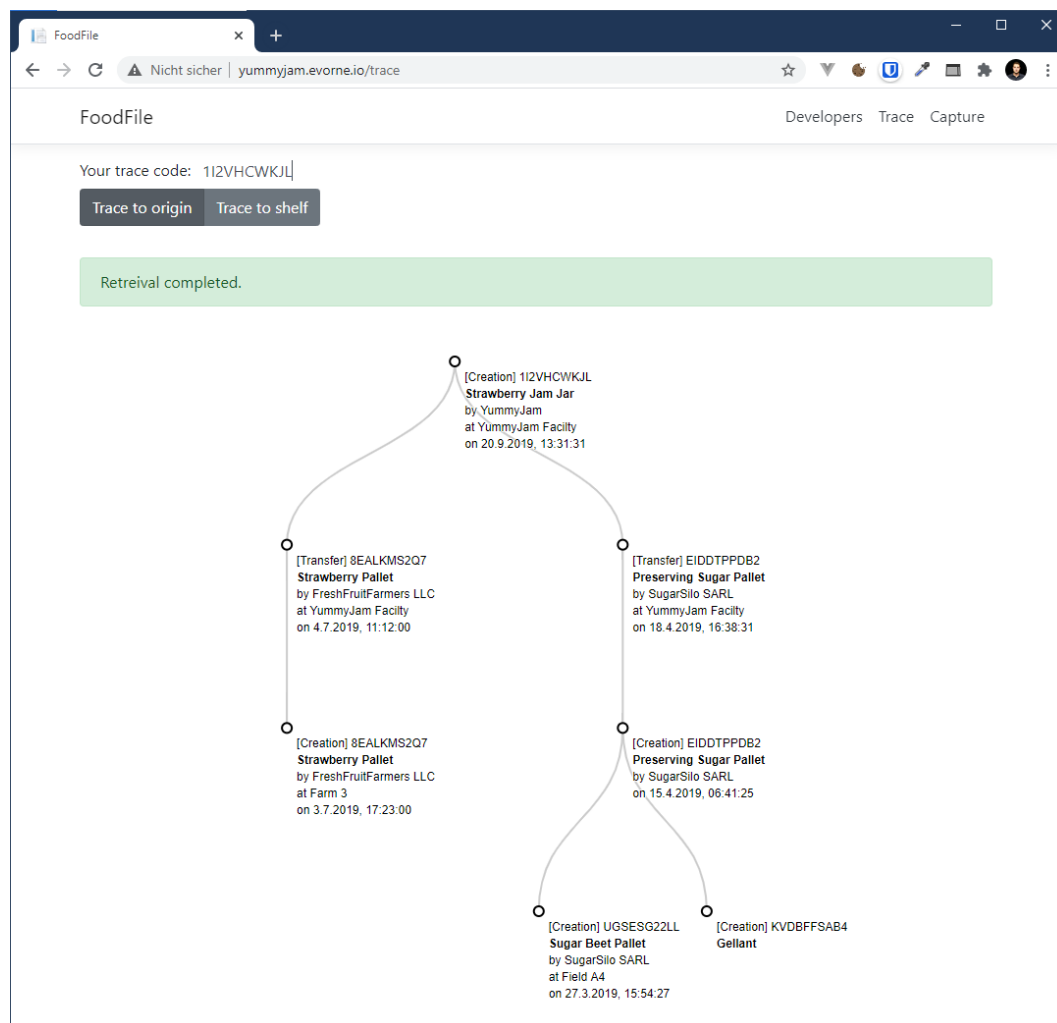


Figure 9: 'Tracing Abilities' scenario result 1

<sup>36</sup> Whether one needs to navigate to the producer instance of an entity to successfully trace it generally depends on the degree of interlinkage, the degree of replication and the distance (in terms of supply chain steps).

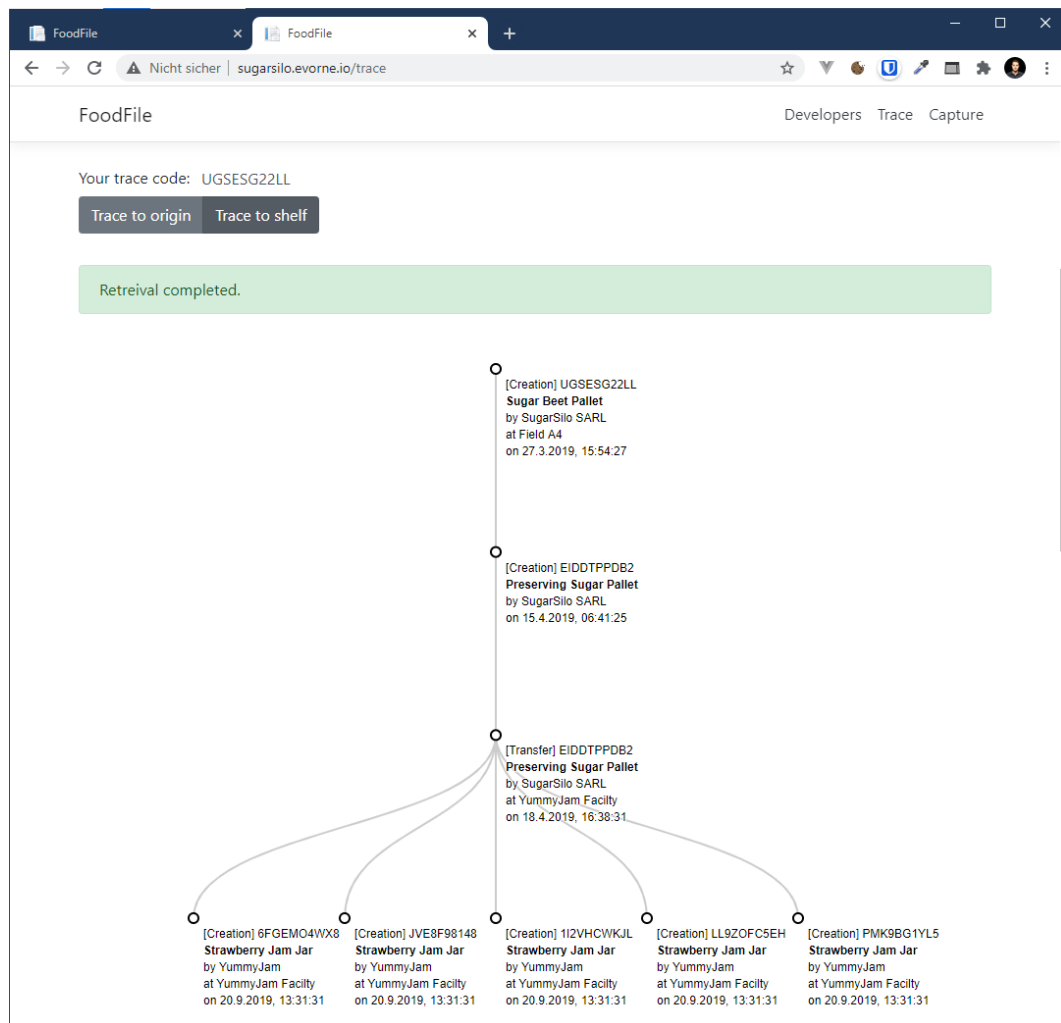


Figure 10: 'Tracing Abilities' scenario result 2

#### 8.4.3 Trace Trigger Perspectives

**Purpose:** The purpose is to show that each supply chain member can trace every entity for which data is locally available and for which the locally available data features member-IDs for follow-up searches.

**Setup:** The setup is equivalent to 8.4.2.

**Scenario library:** `basic_linked`

**Trigger:** This scenario will exemplarily cover three queries that have not been executed in 8.4.2. The respective traces are triggered as in 0, the trace configurations are stated in the Table 9.

**Acceptance criteria:** Table 9 mentions the expected trace results by listing the entity-IDs in the expected-column. A trace result expectation technically needed to be defined on an atom level. Entity level is enough in the case because:

- In the given setup, all atoms for an entity are stored at the same member. Hence it can be expected that they are either retrieved completely or not at all.
- The completeness of the result can still be verified by comparing the screenshots to 8.3.

**Results:** Table 9 shows the tested entity / member / direction combinations, expected results and a reference to the received visualization. The trace results match the expectations.

Entity	Member	Direction	Expected	Received
Strawberry Pallet	FreshFruitFarmers	Downstream	8EALKMS2Q7, all five jam jars	Figure 11
Strawberry Pallet	YummyJam	Upstream	8EALKMS2Q7 (creation & transport)	Figure 12
Preserving Sugar	YummyJam	Upstream	EIDDTPPDB2, UGSESG22LL, KVDBFFSAB4	Figure 13

Table 9: tested combinations

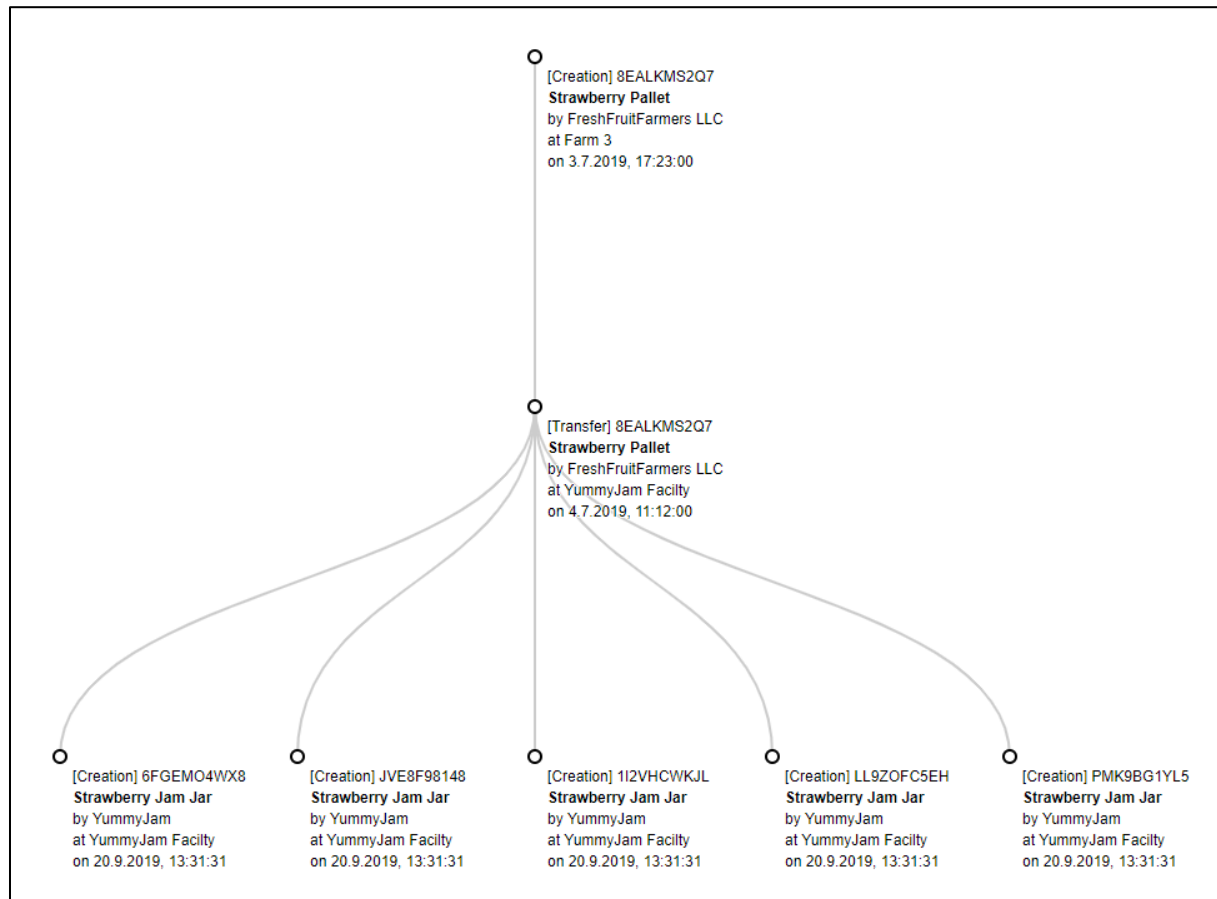


Figure 11: 'Trace Trigger Perspectives' scenario result 1

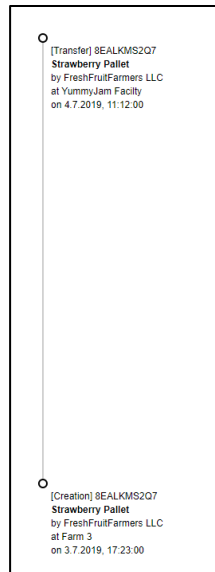


Figure 12:

'Trace Trigger Perspectives' scenario result 2

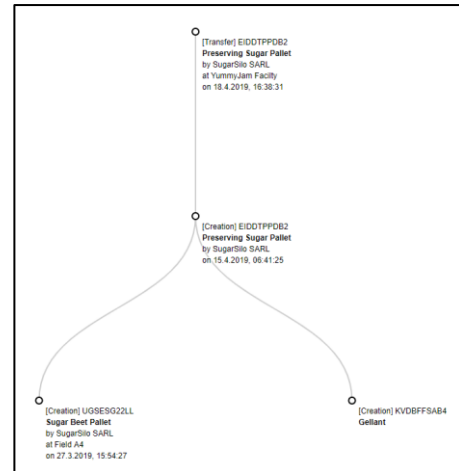


Figure 13:

'Trace Trigger Perspectives' scenario result 3

#### 8.4.4 Retrospective Data Change

**Purpose:** The reality of food production implies that the flow of goods is not always known beforehand. E.g. when tea is harvested on a contractually unbound farm, it is not necessarily clear to whom it will be sold in the future. There might also be cases where supply chain members add or edit data retrospectively. This scenario verifies

- that bidirectional tracing finds all information available disregarding when it was added.
- that it is possible to connect FoodFile to production systems by testing the capture-API.

**Setup:** The same configuration of atoms as in 8.4.2 is shipped during the configuration phase. In addition, the `CapturePrepare` feature of the distribution tool is used to print `st_1_tr_1`, which is an additional transfer step for strawberries, to the console. This output is then added to the `FreshFruitFarmers` instance through the UI of the FoodFile client (Figure 14, Figure 15). The FoodFile client leverages the regular capture-API to push information.

Figure 14: knowledge distribution tool output

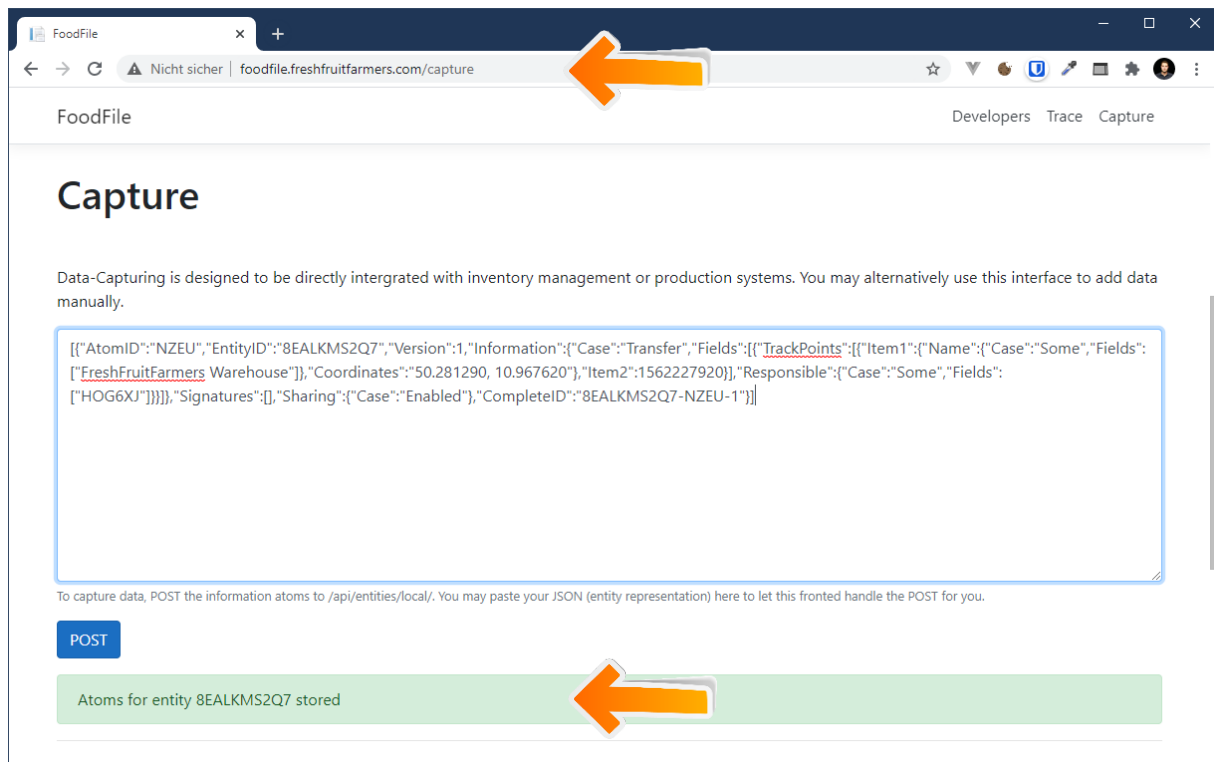


Figure 15: capturing information

**Scenario library:** `added_after`

**Trigger:** To trigger the trace, we execute a 'trace to origin' from the YummyJam instance for one of the five jam jars and a 'trace to shelf' for the strawberries from the FreshFruitFarmers instance.

**Acceptance criteria:**

- The added transfer step is visible in the trace results for upstream- and downstream tracing.

**Results:** Both, upstream and downstream, correctly show the added transfer (Figure 16, Figure 17)

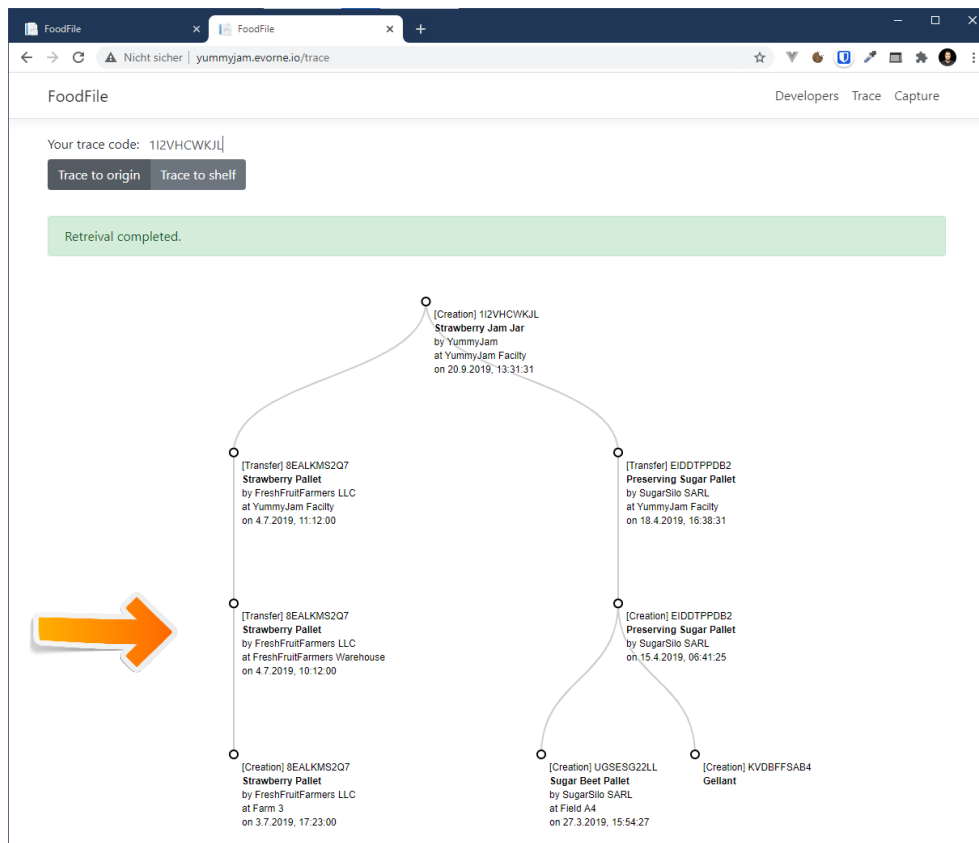


Figure 16: 'Retrospective Data Change' scenario result 1

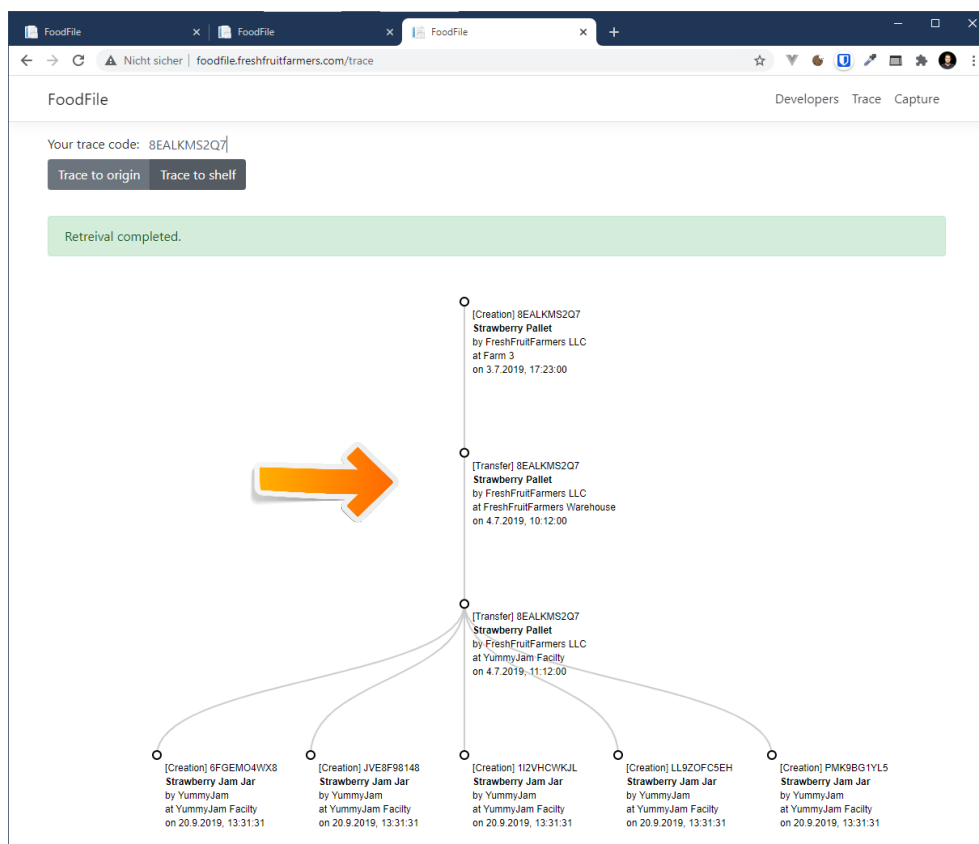


Figure 17: 'Retrospective Data Change' scenario result 2



#### 8.4.5 External Information

**Purpose:** It cannot be assumed every food producer will be part of the FoodFile network. This scenario shows that FoodFile can store information about entities even if the producers themselves are not part of the FoodFile network.

**Setup:** This scenario is based on the atom distribution in 8.4.2 but adds `ge_1_cr_1` to the knowledge base of SugarSilo.

**Scenario library:** `additional_gellant_info`

**Trigger:** An arbitrary upstream search that involves the gellant.

**Acceptance criteria:**

- The trace shows additional information for the Gellant production (when and where it was produced) even though 'The chemical company' does not have a FoodFile member-ID.

**Result:** The acceptance criteria is met (Figure 18).

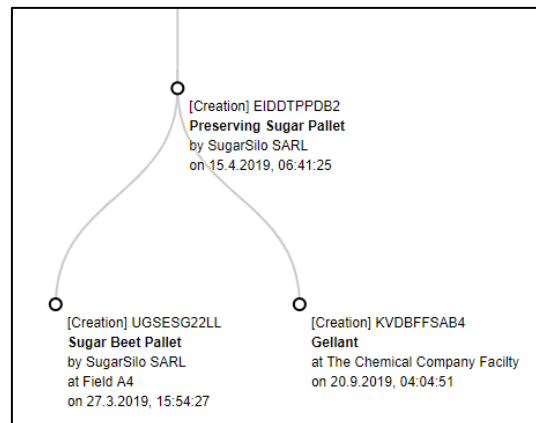


Figure 18: 'External Information' scenario result

#### 8.4.6 Redundancy Advantages

**Purpose:** The Purpose of this scenario is

- to show that FoodFile is resistant against temporary local outages or bankruptcies of members if the participants opt to keep the respective information redundant.
- Redundancy fosters a wider variety of trace options as enables the information maintainer to trace across all locally available information. It specifically allows for downstream traces for ingredients from the perspective of subsequent producers. In this example, it should allow YummyJam to successfully execute a downstream trace for strawberries.

**Setup:** In this scenario YummyJam maintains all atoms that were distributed across the three example members in 8.4.2. The knowledge base of SugarSilo is empty and the knowledge base of FreshFruitFarmers is equal to scenario 8.4.2.

**Scenario library:** `desired_redundancy`

**Trigger:** Three searches are triggered: One downstream trace for strawberries from the perspective of FreshFruitFarmers, one downstream trace for strawberries from the perspective of YummyJam and one upstream trace for a jam jar from the perspective of YummyJam.

**Acceptance Criteria:**

- The visual result for all three traces must not deviate from the visual result seen in 8.4.2.

**Result:** The acceptance criteria is met (Figure 19, Figure 20, Figure 21).

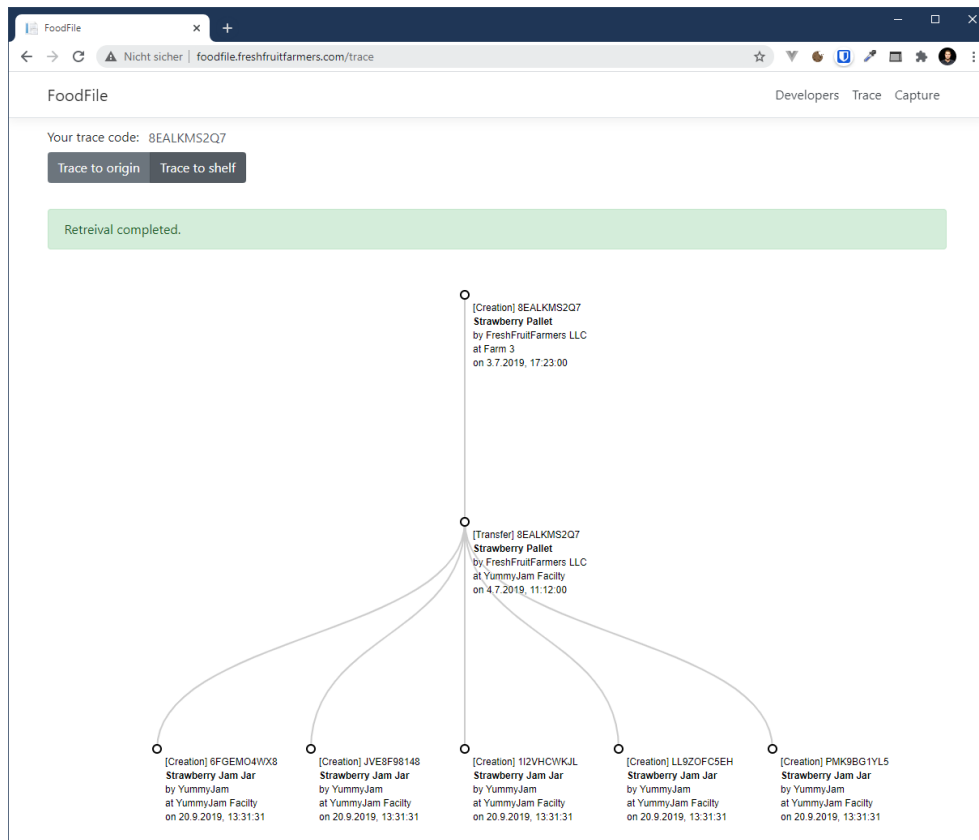


Figure 19: 'Redundancy Advantages' scenario result 1

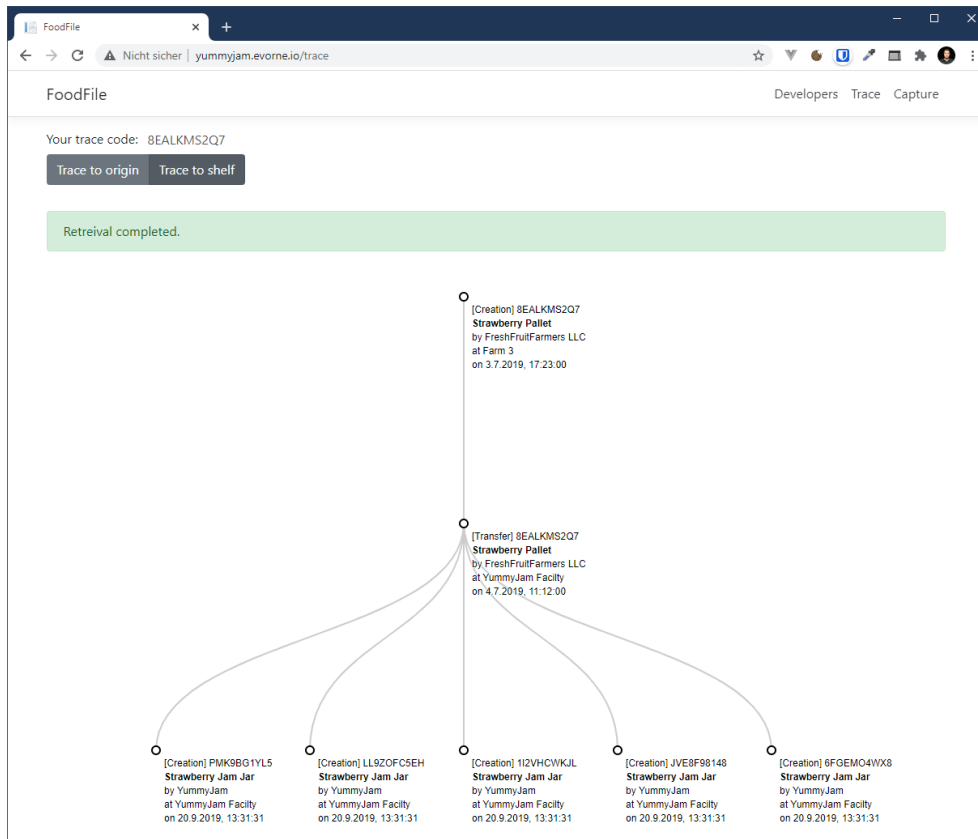


Figure 20: 'Redundancy Advantages' scenario result 2

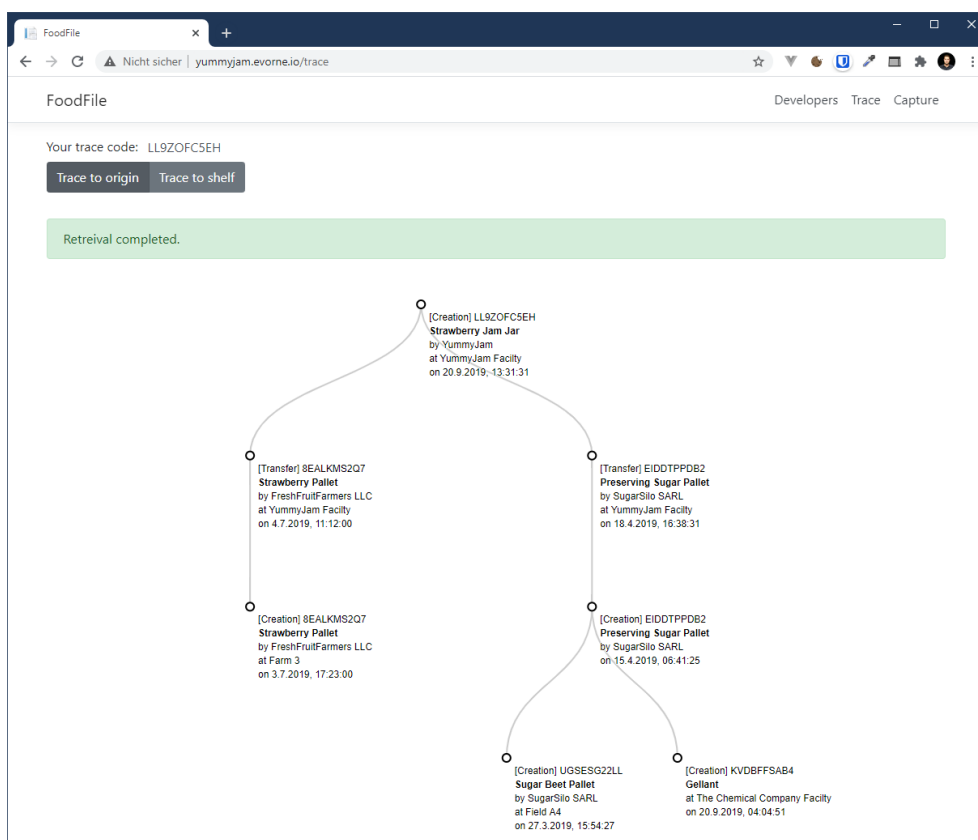


Figure 21: 'Redundancy Advantages' scenario result 3

### 8.4.7 Incomplete Information

**Purpose:** This scenario verifies that the system can cope with incomplete information by successfully completing a trace though datapoints are missing.

**Setup:** The setup is equivalent to 0, but without any transfer atoms.

**Scenario library:** `no_transport`

**Trigger:** An upstream trace for a jam jar from the YummyJam instance.

**Acceptance criteria:**

- The preserving sugar and the gellant creation are correctly displayed even though the preserving sugar does not have a location information and the gellant is missing in-bound entities, location and time of creation.
- While transportation information is missing, the tree of ingredients remains traceable.

**Result:** The acceptance criteria is met (Figure 22, Figure 23).

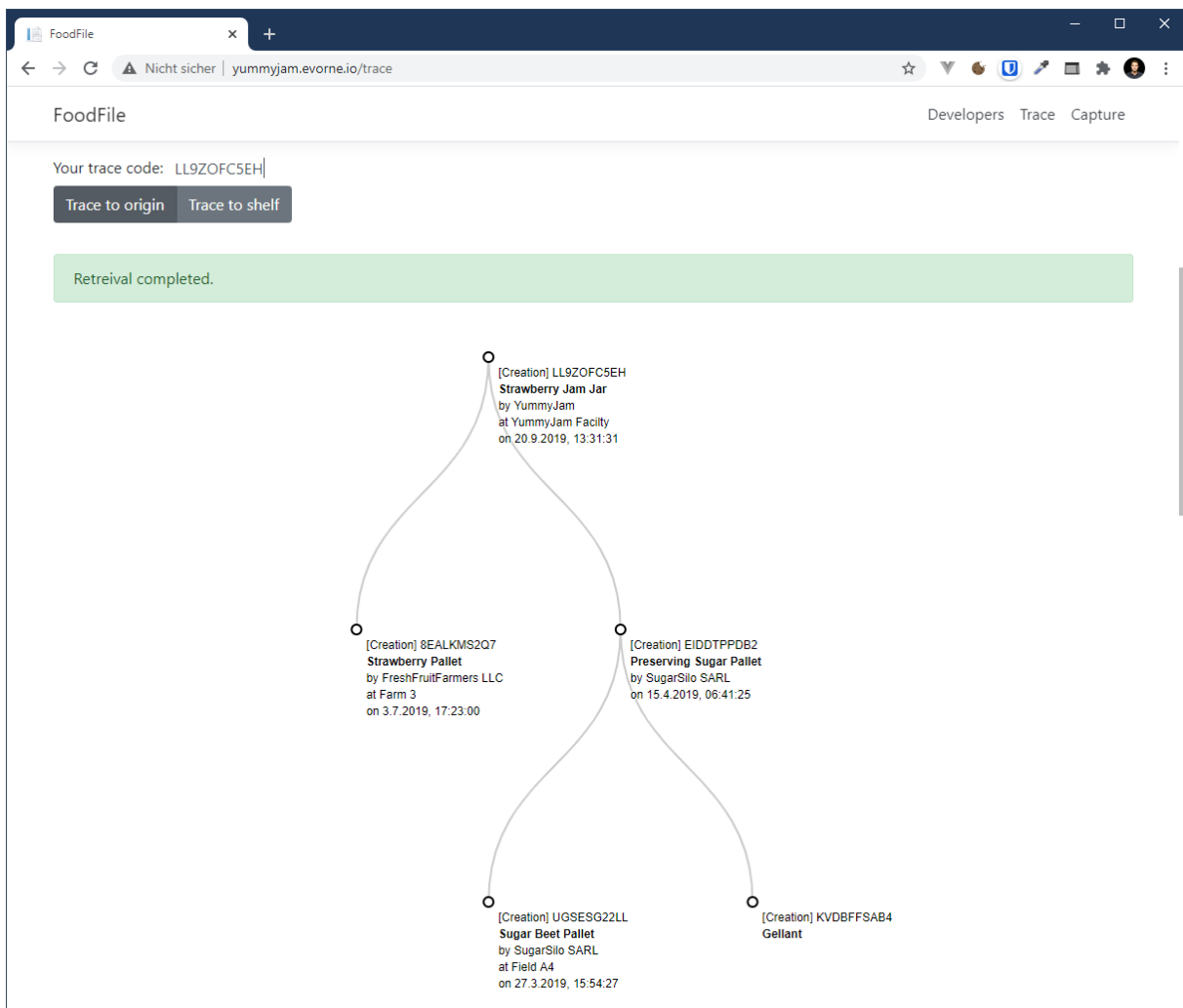


Figure 22: 'Incomplete Information' scenario result

### 8.4.8 Incorrect Information

**Purpose:** This test should verify that the system can cope with incorrect information, i.e. incorrect member-IDs. As a successful trace depends on the possibility to identify the correct members for follow up queries, the desired behavior is that all trace paths which are not affected by the incorrect member-ID are displayed correctly.

**Setup:** The setup is as in 8.4.2. The only difference is that in the knowledge base of YummyJam, the involvement atom pointing to FreshFruitFarmers is replaced by an involvement atom with an invalid member-ID (jj\_2\_iv\_2).

**Scenario library:** `unknown_strawberry_producer`

**Trigger:** An upstream trace for a jam jar from the perspective of YummyJam.

**Acceptance criteria:**

- The preserving sugar branch must be equal to the preserving sugar branch in 8.4.2.
- The Strawberry branch expected to show an inferred creation atom without any further information.

**Result:** The acceptance criteria is met (Figure 23).

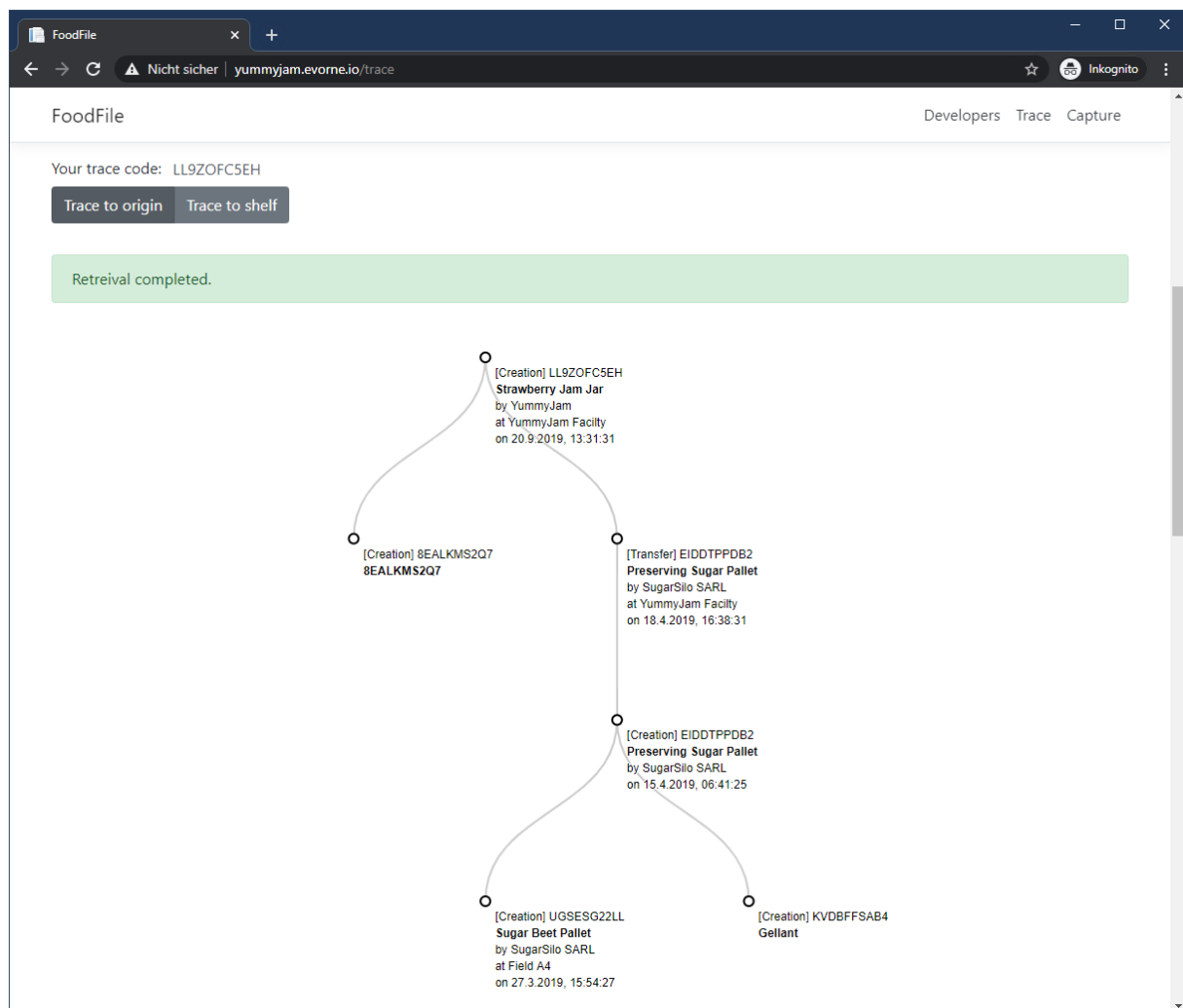


Figure 23: 'Incorrect Information' scenario result

#### 8.4.9 Consumer Accessibility

**Purpose:** This test verifies that information is accessible for non-participants.

**Setup:** The setup is equal to 8.4.2.

**Scenario library:** `basic_linked`

**Trigger:** An arbitrary (here: Strawberries downstream) trace executed from an instance that has a member-ID nor a database associated. For this test, the server instance is hosted on `localhost:80` and the SPA connecting to it on `localhost:5001`. FoodFile is programmed to show the associated member-ID in the lower right corner (footer) of the page. This can be seen in screenshots of the deployment scenario (8.4.1)<sup>37</sup>.

The trace is triggered by navigating to `localhost:5001/trace`, selecting trace to shelf and using the following search term: `HOG6XJ-8EALKMS2Q7`. The first block of the search-term denotes the member-ID of FreshFruitFarmers, the second block is the entity-ID of the Strawberries.

**Acceptance criteria:**

- The instance used for this test neither has a member-ID nor a database
- The Gellant trace shows all expected information in accordance with the exemplary supply chain (8.3).

**Results:** At first, a trace is triggered for `8EALKMS2Q7` (without member prefix) to show that there is no local information available and to show that the local instance does not have a member-ID (Figure 24). The arrows indicate the missing results (no visualization) and the missing member information. Then, a second trace is executed as described in the trigger-section above (Figure 25).

---

<sup>37</sup> In most screenshots of this evaluation, the footer is not visible as it is outside the browser's viewport.

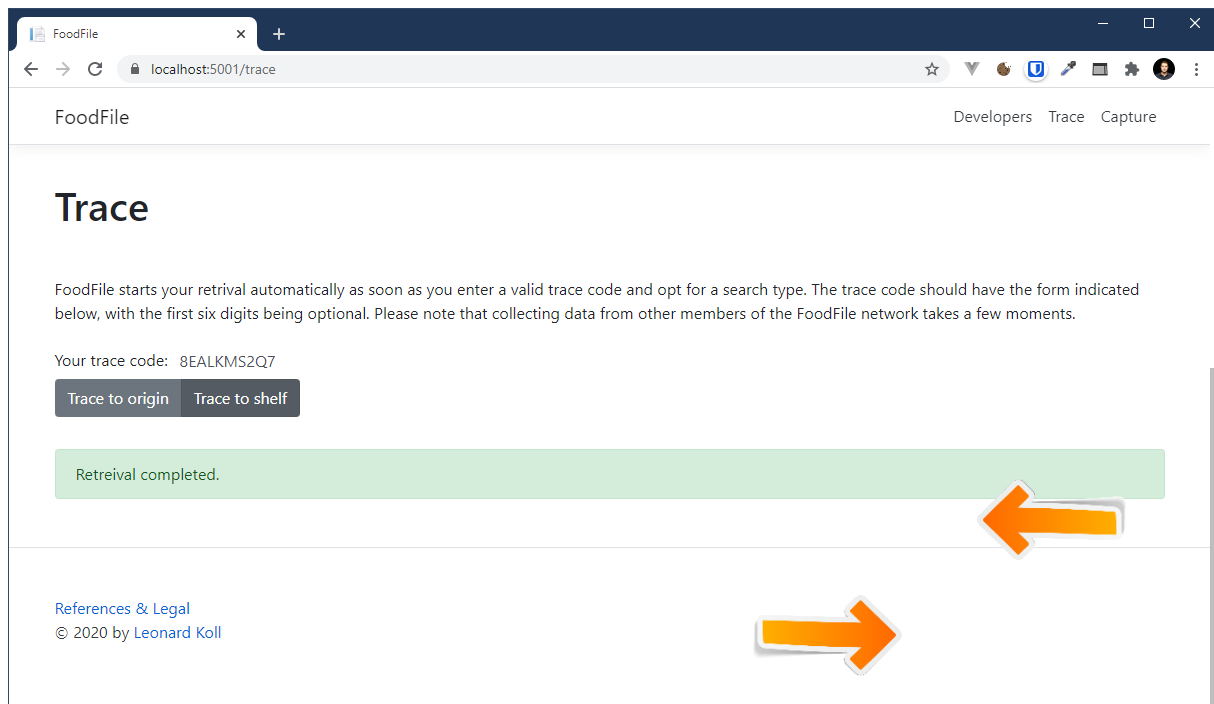


Figure 24: 'Consumer Accessibility' scenario result 1

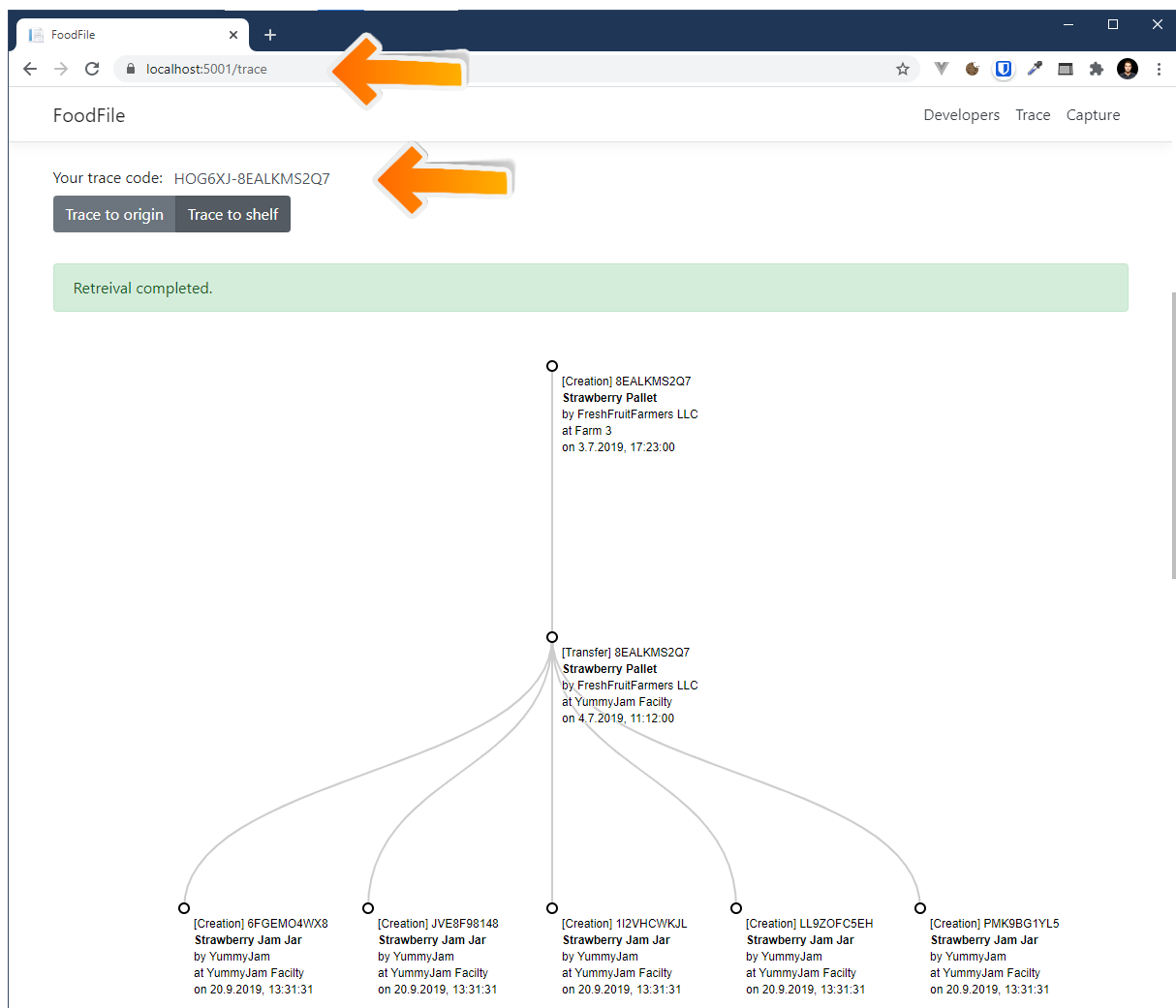


Figure 25: 'Consumer Accessibility' scenario result 2

#### 8.4.10 Token Protection

**Purpose:** This test verifies that information sharing regulation on atom level is possible and effective.

**Setup:** The scenario is based on added after with the difference that the additional strawberry transfer (`st_1_tr_1`) is added by the distribution tool instead of through the API and that the atom has the sharing policy set to `ByToken`.

**Scenario library:** `token_protected`

**Trigger:** One strawberry upstream trace from the perspective of YummyJam without token and one trace with token.

**Acceptance criteria:**

- The additional transfer must not be visible in the trace without token (triggered with the search term `HOG6XJ-8EALKMS2Q7`)
- The additional transfer must be visible in the trace with token (triggered with the search term `HOG6XJ-8EALKMS2Q7-24c1868b13ebf1afc694a70291ffd73e8ba50671eb81ff4af678da5aa683623e`)

`HOG6XJ` is the member-ID of FreshFruitFarmers, `8EALKMS2Q7` is the entity-ID of the strawberries and the long alphanumeric token is the result of the SHA256 function applied on the entity-ID concatenated with the FreshFruitFarmers secret, which in this example is 'salt' (the default).

$$SHA256(8EALKMS2Q7salt) = 24c1 \dots 623e$$

**Result:** The acceptance criteria is met (Figure 26, Figure 27).



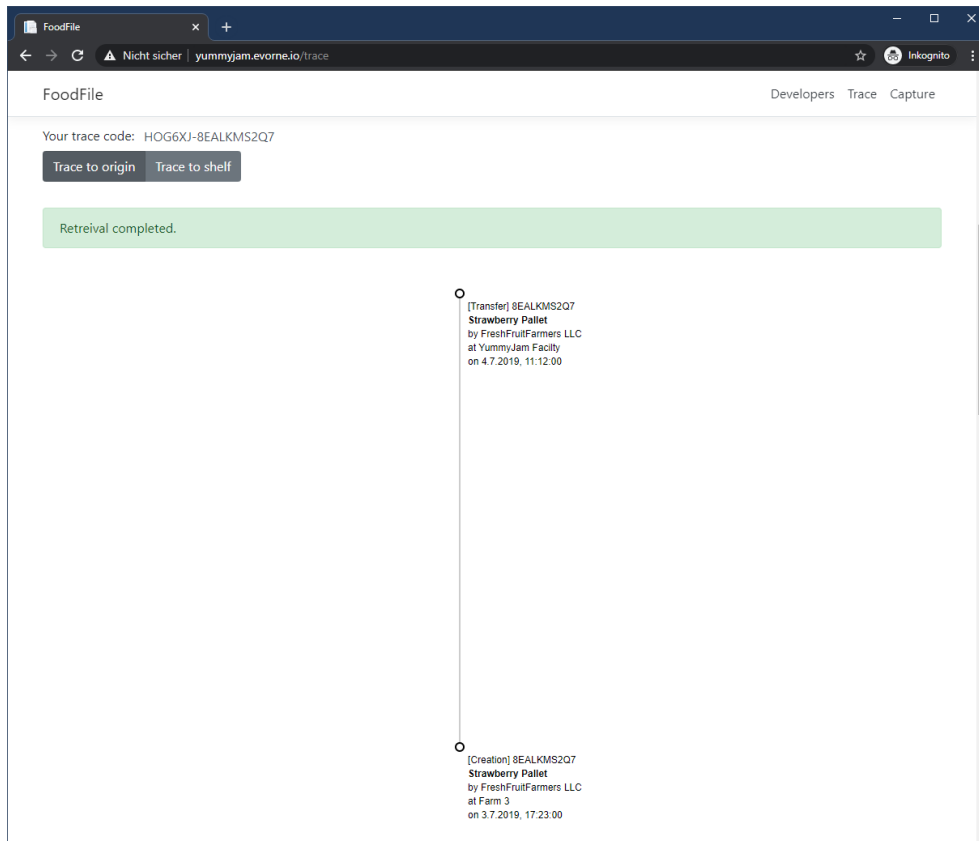


Figure 26: 'Token Protection' scenario result 1

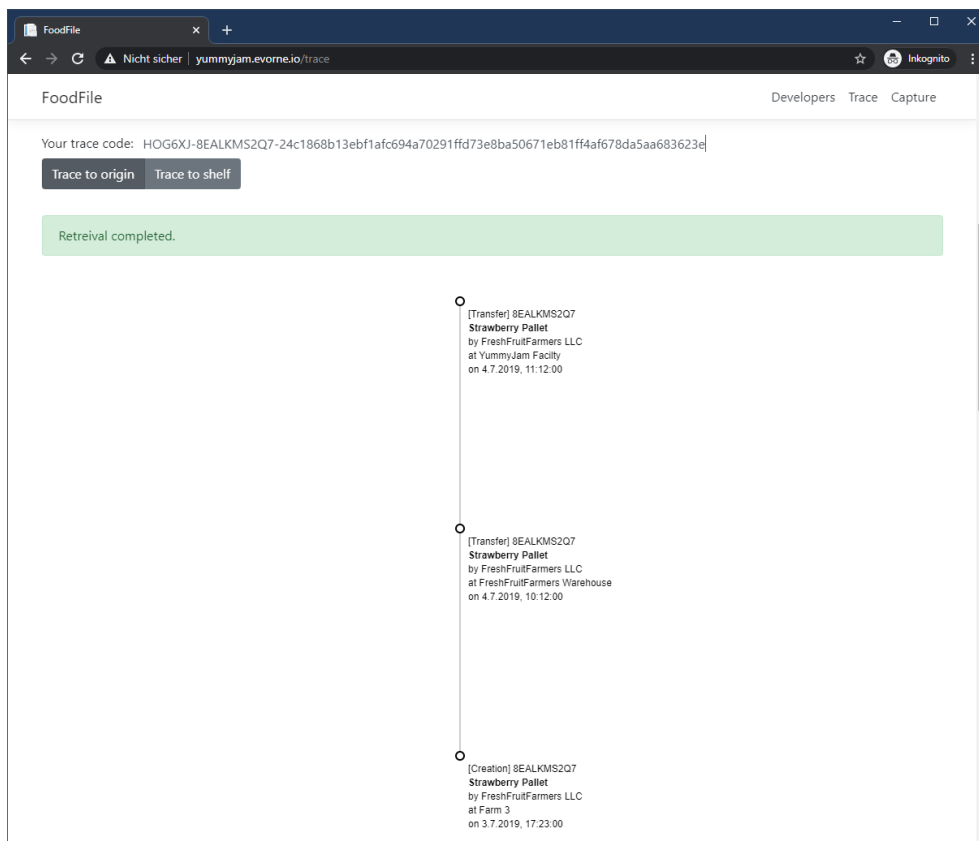


Figure 27: 'Token Protection' scenario result 2

### 8.4.11 Replication

**Purpose:** This test verifies that information replication is possible to the degree desired by participants.

**Setup:** The setup is equal to Table 8, without the involvement atoms.

**Scenario library:** `basic`

**Trigger:** After knowledge distribution, an upstream trace for a jam jar is triggered from the perspective of YummyJam. Then, the fetch feature is used on the YummyJam instance to copy information from the FreshFruitFarmers and SugarSilo instances and hence create redundancies. The fetch feature can be found in the Capture section of the SPA. It takes a comma separated list of member-IDs specifying the members of which the data should be copied. The same trace as before is executed again.

**Acceptance criteria:**

- The first trace must not show any information beyond the value creation achieved by YummyJam: Since the involvement atoms are missing, the YummyJam instance does not know about any members it could ask for follow-up queries.
- The second trace must show the full trace result as described in 8.3, since the data is now locally available on the instance of YummyJam.

**Result:** Figure 28 shows the trace result before the fetch, Figure 29 shows the fetch and Figure 30 shows the trace result after the fetch. The acceptance criteria are met.

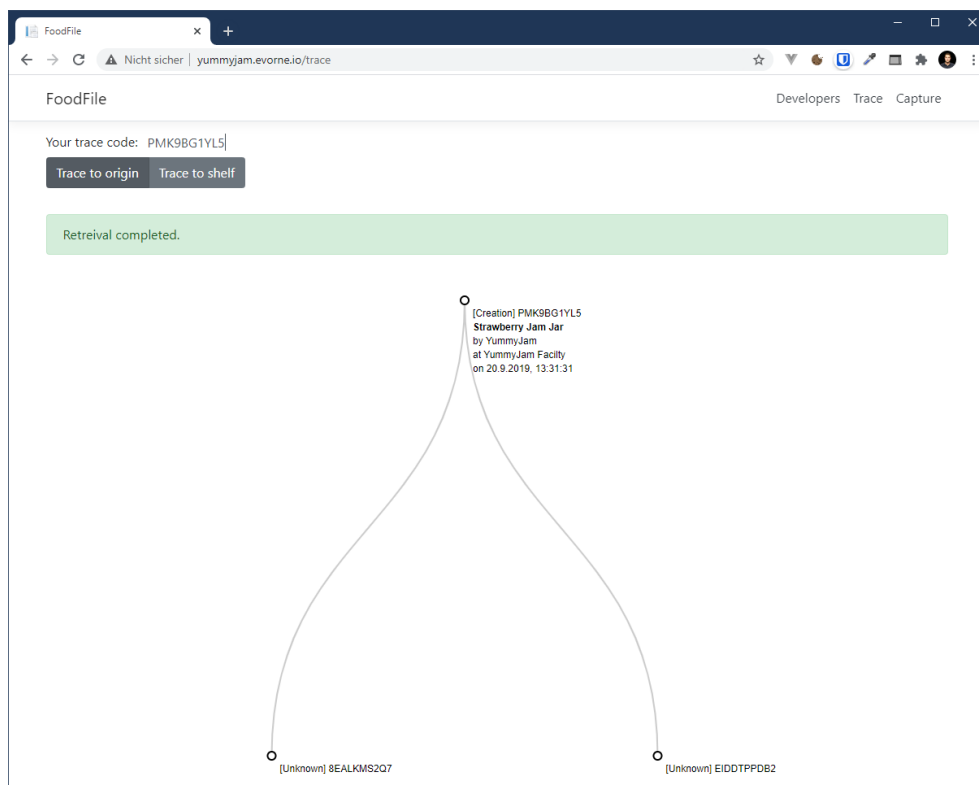


Figure 28: 'Replication' scenario result 1

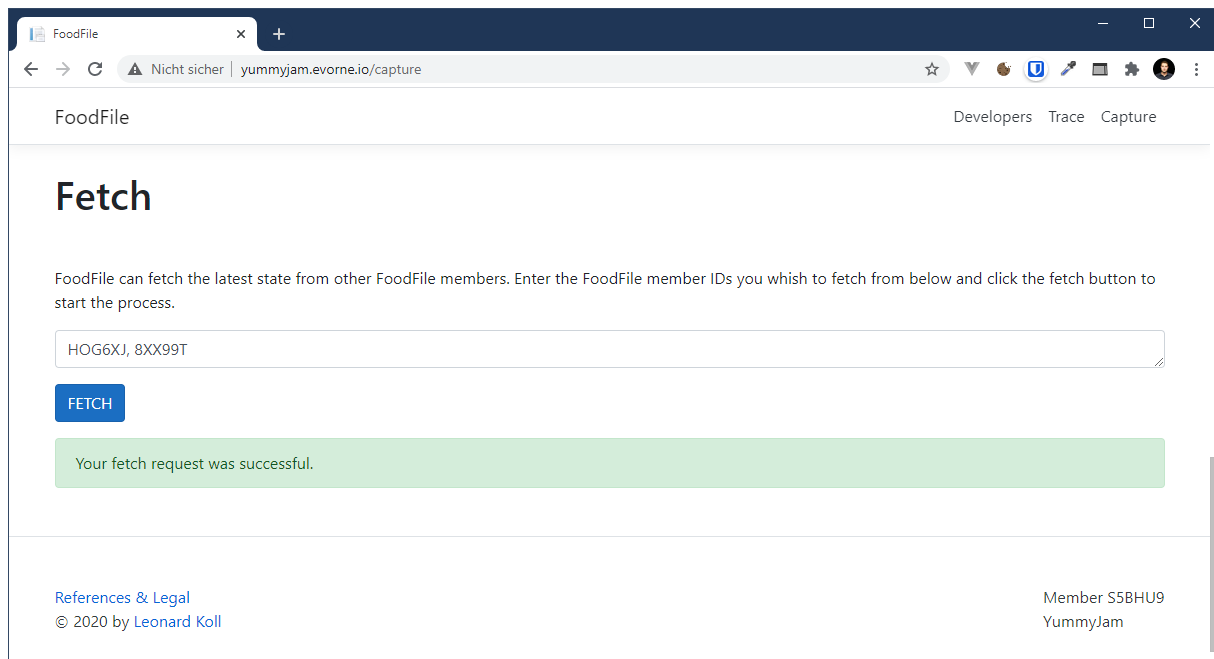


Figure 29: 'Replication' scenario result 2

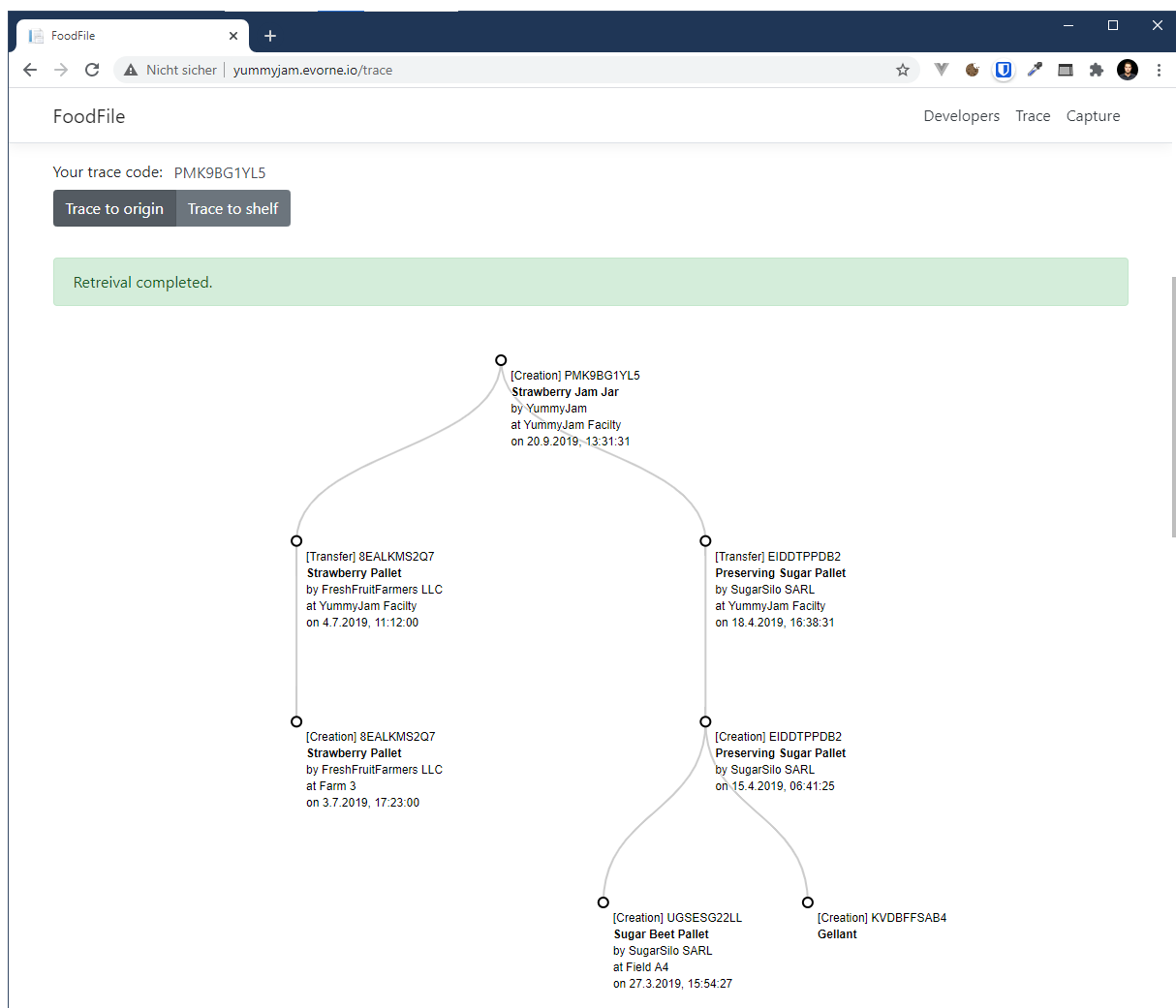


Figure 30: 'Replication' scenario result 3

#### 8.4.12 Contradicting Information

**Purpose:** This test verifies that contradicting information in the FoodFile system is detectable.

**Setup:** The knowledge distribution is based on the scenario described in 8.4.2. In addition, SugarSilo and FreshFruitFarmers each receive a creation atom about the first jam jar, one of which is in line with what is stored on the YummyJam instance (`jj_1_cr_1`, SugarSilo) and one which does not have any inbound entities (`jj_1_cr_2`, FreshFruitFarmers). The ID of the contradiction atom is `6FGEMO4WX8-ZSOC-1` (the final -1 indicates the version).

**Scenario library:** `basic_contradiction`

**Trigger:** An upstream trace for the first jam jar is triggered. Below the trace result visualization there is a list of all atoms delivered as trace result. By clicking verify, the backend requests hashes for this atom (this atom version) from all members involved in the search. The hashes are collected and summarized. The number in brackets (herein after called support count) indicates how many members replied with this hash.

**Acceptance criteria:**

- The support count for atom `6FGEMO4WX8-ZSOC-1` is divided in two hashes with a support count of two and one (which is the one without inbound entities).
- The support count for all other atoms is 1, as there is only one copy per atom in the system.

**Result:** Figure 31 shows the trace result and the retrieved support counts.

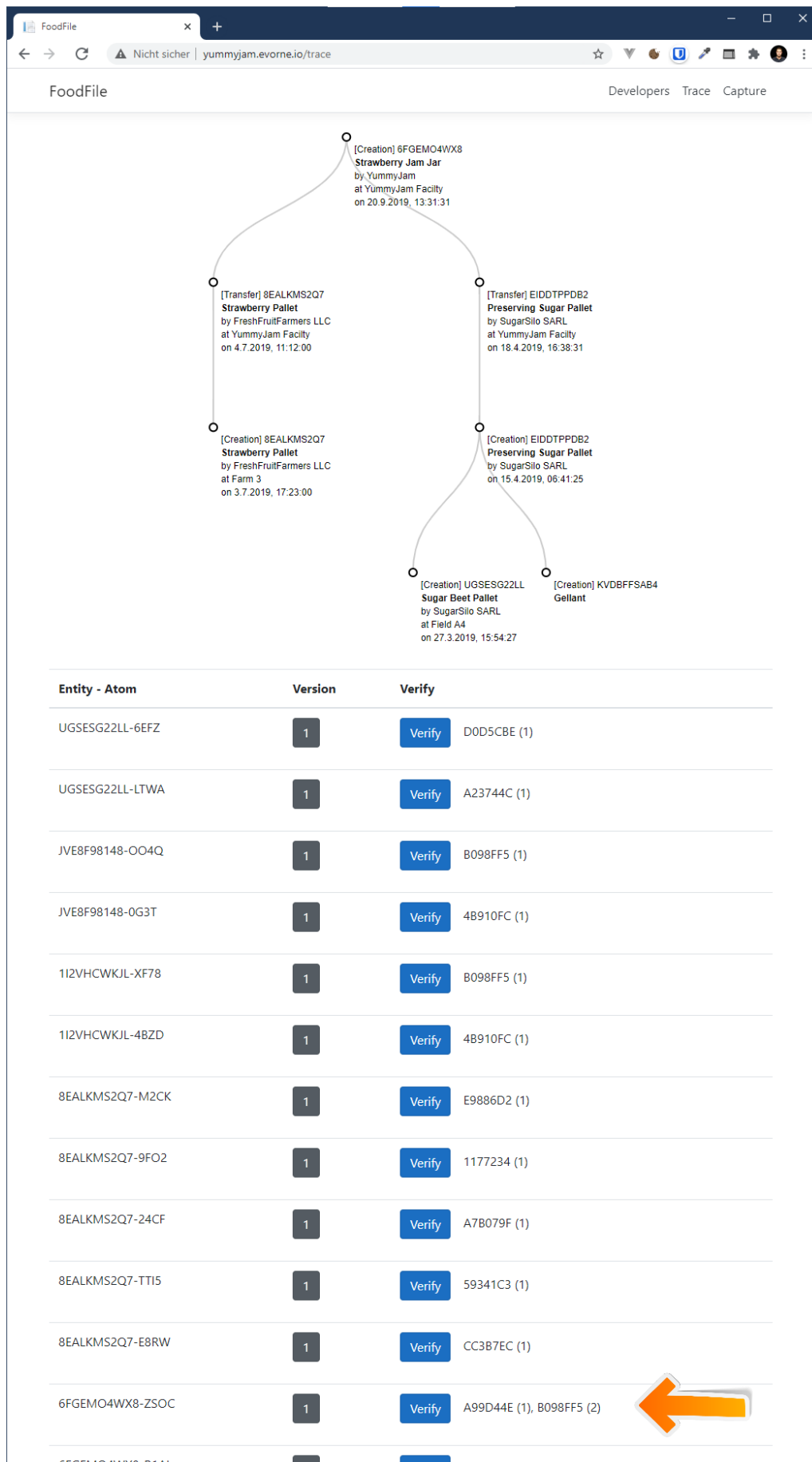


Figure 31: 'Contradicting Information' scenario result

### 8.4.13 Contradicting Information Versions

**Purpose:** This scenario shows that the atom verification is applied on an atom version level: One version of an atom can be contradicting while another version of the same atom can be aligned. In practice, this may occur when a conflict is detected and resolved through human interaction.

**Setup:** The setup adds to 8.4.12: An additional version of 6FGEMO4WX8-ZSOC (jj\_1\_cr\_3) is introduced which is aligned across FreshFruitFarmers, SugarSilo and YummyJam.

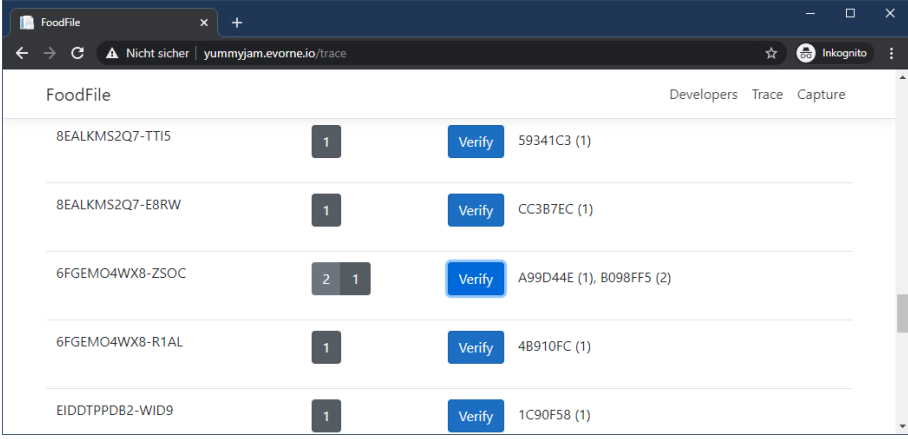
**Scenario library:** `version_contradiction`

**Trigger:** A downstream trace for jam jar 1 followed by triggering the verification procedure for the received atoms. The Verify button (and the result next to it) relate to the selected version in the version column.

**Acceptance criteria:**

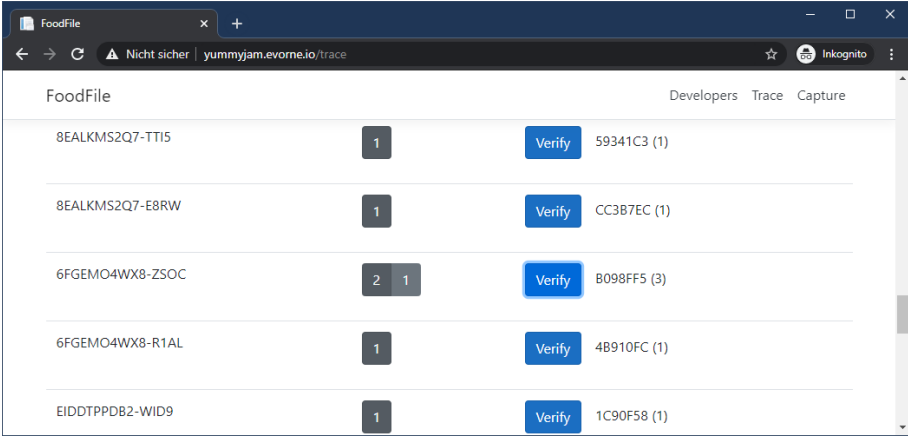
- For 6FGEMO4WX8-ZSOC-1, the support counts must equal the results from 8.4.12.
- For 6FGEMO4WX8-ZSOC-2, all members must report the same hash.

**Results:** Figure 32 shows the support counts for version 1, Figure 33 shows the support count for version 2.



FoodFile			
			Developers Trace Capture
8EALKMS2Q7-TT15	1	Verify	59341C3 (1)
8EALKMS2Q7-E8RW	1	Verify	CC3B7EC (1)
6FGEMO4WX8-ZSOC	2 1	Verify	A99D44E (1), B098FF5 (2)
6FGEMO4WX8-R1AL	1	Verify	4B910FC (1)
EIDDTPPDB2-WID9	1	Verify	1C90F58 (1)

Figure 32: 'Contradicting Information Versions' scenario result 1



FoodFile			
			Developers Trace Capture
8EALKMS2Q7-TT15	1	Verify	59341C3 (1)
8EALKMS2Q7-E8RW	1	Verify	CC3B7EC (1)
6FGEMO4WX8-ZSOC	2 1	Verify	B098FF5 (3)
6FGEMO4WX8-R1AL	1	Verify	4B910FC (1)
EIDDTPPDB2-WID9	1	Verify	1C90F58 (1)

Figure 33: 'Contradicting Information Versions' scenario result 2

#### 8.4.14 Resilient Histogramy (1)

**Purpose:** This scenario is the first of three scenarios verifying a resilient histogramy of data changes. The test analyzes the system's behavior when a malicious participant modifies an existing atom version.

**Setup:** In this scenario, SugarSilo adjusts the production date of the gellant to be later than what was logged originally. The adjustment is done by modifying an existing atom version. As modifications are only allowed through auditing (by creating a new version), SugarSilo acts maliciously. All three members have the same knowledge base except two atoms which deviate in the knowledge base of SugarSilo: The creation atom (version 1 & 2) each mention a different production date (ge\_1\_cr\_3 and ge\_1\_cr\_4 instead of ge\_1\_cr\_1 and ge\_1\_cr\_2). The ID of the manipulated atom is KVDBFFSAB4-IFG2.

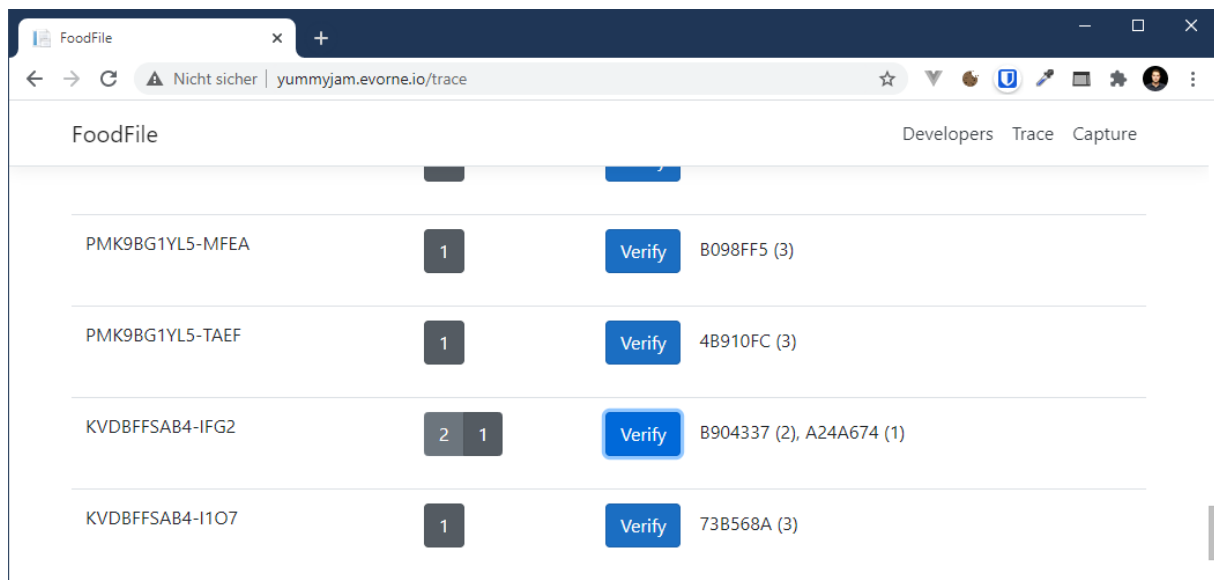
**Scenario library:** retrospective\_change\_after

**Trigger:** An upstream search for a jam jar from the perspective of YummyJam.

**Acceptance criteria:**

- The manipulation must be uncoverable by inspecting the support counts for the atom in question. Both atom versions which have been manipulated must show a respective support count of two members versus one member.

**Result:** Figure 34 shows the support count for version 1, Figure 35 shows the support count for version 2.



The screenshot shows a web browser window with the URL 'yummyjam.evorne.io/trace'. The page title is 'FoodFile' and it has tabs for 'Developers', 'Trace', and 'Capture'. The main content is a table with four rows of atom data. Each row contains an atom ID, a support count (represented by a grey box with a number), a 'Verify' button, and a list of supporting members in parentheses.

Atom ID	Support Count	Verify	Supporting Members
PMK9BG1YL5-MFEA	1	Verify	B098FF5 (3)
PMK9BG1YL5-TAEF	1	Verify	4B910FC (3)
KVDBFFSAB4-IFG2	2, 1	Verify	B904337 (2), A24A674 (1)
KVDBFFSAB4-I1O7	1	Verify	73B568A (3)

Figure 34: 'Resilient Histogramy (1)' scenario result 1

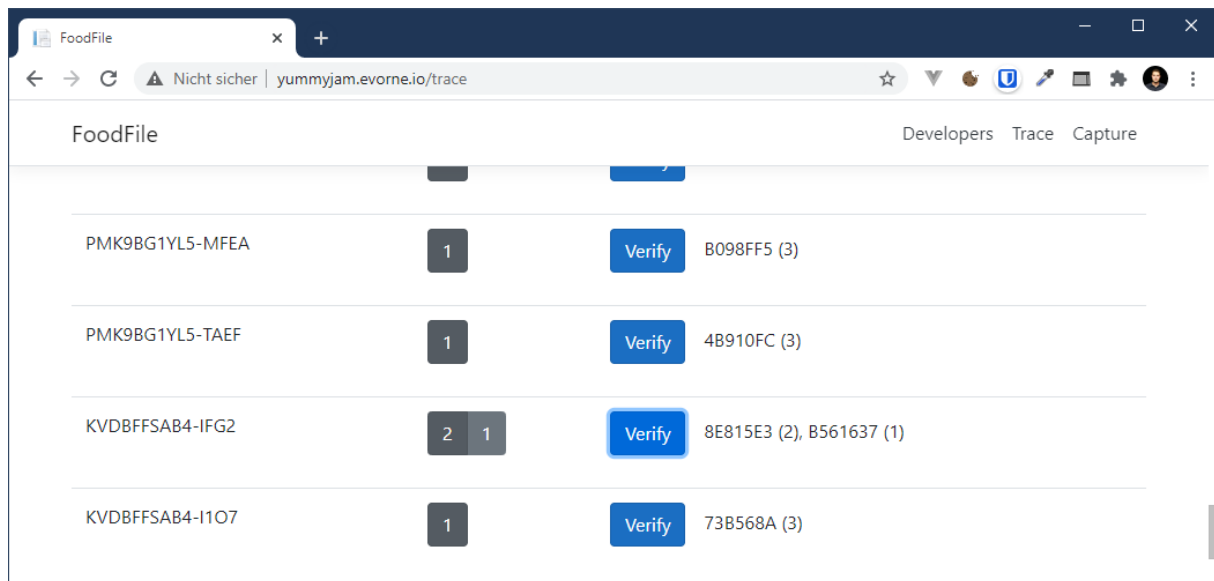


Figure 35: 'Resilient Histogramy (1)' scenario result 2

#### 8.4.15 Resilient Histogramy (2)

**Purpose:** This scenario is the second of three scenarios verifying a resilient histogramy of data changes. The test analyzes the system's behavior when a malicious participant modifies information by doing both: First, creating a new atom version with an increased number and second, deleting the previous versions from its knowledge base.

**Setup:** In this scenario, SugarSilo manipulates the production date to be later than what was logged originally. As opposed to the configuration in 8.4.14, SugarSilo veils this activity by creating a new version (higher version number) and deletes the previous versions from its knowledge base (ge\_1\_cr\_5 instead of ge\_1\_cr\_1 and ge\_1\_cr\_2). This scenario assumes that other FoodFile members do not agree with the changes made and hence do not adopt it in their knowledge base.

The knowledge bases of FreshFruitFarmers and YummyJam are equal; the knowledge base of SugarSilo deviates as described in the purpose section. The ID of the manipulated atom is KVDBFFSAB4-IFG2.

**Scenario library:** retrospective\_change\_version

**Trigger:** An upstream search for a jam jar from the perspective of YummyJam.

**Acceptance criteria:**

- The questionable modification must be uncoverable by inspecting the support counts for the respective atom versions. The original atom history is expected to have a support count of two, whereas the new version is expected to have a support count of one.

**Result:** Figure 36 shows the support count for version 1, Figure 37 shows the support count for version 2 and Figure 38 shows the support count for the questionable version 3.



FoodFile		Developers	Trace	Capture
LL9ZOFCS5EH-IJ89	1	Verify	4B910FC (3)	
PMK9BG1YL5-MFEA	1	Verify	B098FF5 (3)	
PMK9BG1YL5-TAEF	1	Verify	4B910FC (3)	
KVDBFFSAB4-IFG2	3 2 1	Verify	B904337 (2)	
KVDBFFSAB4-I1O7	1	Verify	73B568A (3)	

Figure 36: 'Resilient Histogramy (2)' scenario result 1

FoodFile		Developers	Trace	Capture
LL9ZOFCS5EH-IJ89	1	Verify	4B910FC (3)	
PMK9BG1YL5-MFEA	1	Verify	B098FF5 (3)	
PMK9BG1YL5-TAEF	1	Verify	4B910FC (3)	
KVDBFFSAB4-IFG2	3 2 1	Verify	8E815E3 (2)	
KVDBFFSAB4-I1O7	1	Verify	73B568A (3)	

Figure 37: 'Resilient Histogramy (2)' scenario result 2

FoodFile		Developers	Trace	Capture
LL9ZOFCS5EH-IJ89	1	Verify	4B910FC (3)	
PMK9BG1YL5-MFEA	1	Verify	B098FF5 (3)	
PMK9BG1YL5-TAEF	1	Verify	4B910FC (3)	
KVDBFFSAB4-IFG2	3 2 1	Verify	B561637 (1)	
KVDBFFSAB4-I1O7	1	Verify	73B568A (3)	

Figure 38: 'Resilient Histogramy (2)' scenario result 3

#### 8.4.16 Resilient Histogramy (3)

**Purpose:** This scenario is the third of three scenarios verifying a resilient histogramy of data changes. In this scenario, the modifications are logged as a new version and spread and accepted across the network. The changes are uncoverable in the version history of the atom and the agreement reflects in the support count.

**Setup:** All three members have the exact same knowledge base. For the creation of gellant, it mentions a change (adding a production-responsible member) in the form of a new atom version (ge\_1\_cr\_1 and ge\_1\_cr\_2).

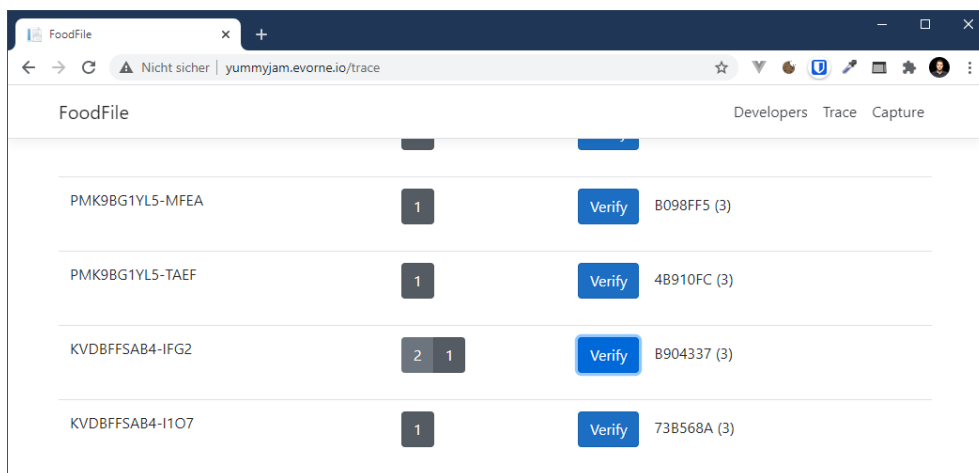
**Scenario library:** `retrospective_change_before`

**Trigger:** An upstream search for a jam jar from the perspective of YummyJam.

**Acceptance criteria:**

- The modification is logged as two atom versions which have a support count of three each.

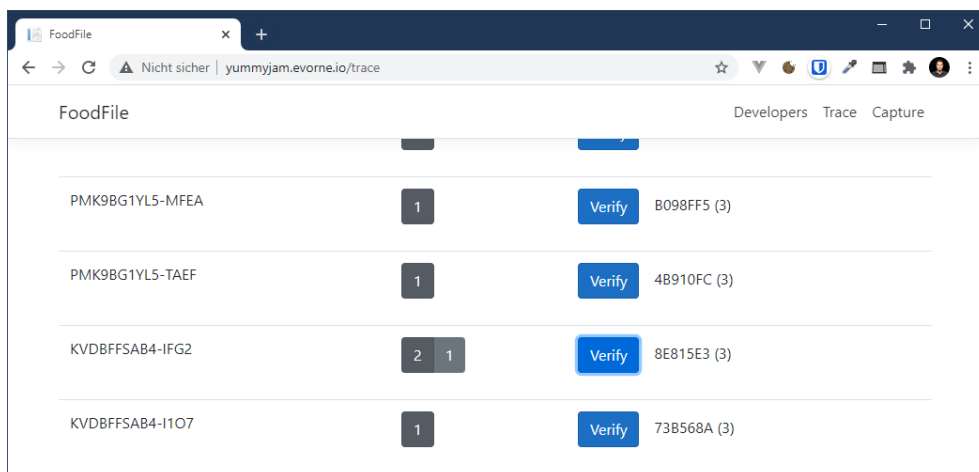
**Result:** Figure 39 shows the support count for version 1, Figure 40 shows the support count for version 2.



The screenshot shows the FoodFile web interface with the URL `yummyjam.evorne.io/trace`. It displays a table of atoms with their support counts for version 1. The atoms are PMK9BG1YL5-MFEA, PMK9BG1YL5-TAEF, KVDBFFSAB4-IFG2, and KVDBFFSAB4-I1O7. The support counts are 1, 1, 2, and 1 respectively. Each row has a 'Verify' button and a hash value in parentheses.

Atom ID	Support Count	Verify	Hash
PMK9BG1YL5-MFEA	1	Verify	B098FF5 (3)
PMK9BG1YL5-TAEF	1	Verify	4B910FC (3)
KVDBFFSAB4-IFG2	2	Verify	B904337 (3)
KVDBFFSAB4-I1O7	1	Verify	73B568A (3)

Figure 39: 'Resilient Histogramy (3)' scenario result 1



The screenshot shows the FoodFile web interface with the URL `yummyjam.evorne.io/trace`. It displays a table of atoms with their support counts for version 2. The atoms are PMK9BG1YL5-MFEA, PMK9BG1YL5-TAEF, KVDBFFSAB4-IFG2, and KVDBFFSAB4-I1O7. The support counts are 1, 1, 2, and 1 respectively. Each row has a 'Verify' button and a hash value in parentheses.

Atom ID	Support Count	Verify	Hash
PMK9BG1YL5-MFEA	1	Verify	B098FF5 (3)
PMK9BG1YL5-TAEF	1	Verify	4B910FC (3)
KVDBFFSAB4-IFG2	2	Verify	8E815E3 (3)
KVDBFFSAB4-I1O7	1	Verify	73B568A (3)

Figure 40: 'Resilient Histogramy (3)' scenario result 2

### 8.4.17 Histography Visualization

**Purpose:** This scenario demos the ability of the graphical user interface to automatically update the tree visualization of the trace when the user selects a different atom version.

**Setup:** The knowledge base distribution is equal to the configuration in 8.4.16.

**Scenario library:** `retrospective_change_before`

**Trigger:** A downstream trace for jam jar 1 followed by a switch from KVDBFFSAB4-IFG2-2 to KVDBFFSAB4-IFG2-1.

**Acceptance criteria:**

- The visualization must adapt once the selected atom version changes. Version 2 must show a responsible member-ID for the gellant production while version 1 must not show this information.

**Result:** Figure 41 shows a side by side comparison of the same trace with the two different versions of KVDBFFSAB4-IFG2 selected. To improve readability, all atoms above the remaining three visible ones have been removed from the DOM retrospectively through the browser's development console. Version 2 shows a responsible member-ID for the gellant production, version 1 does not.

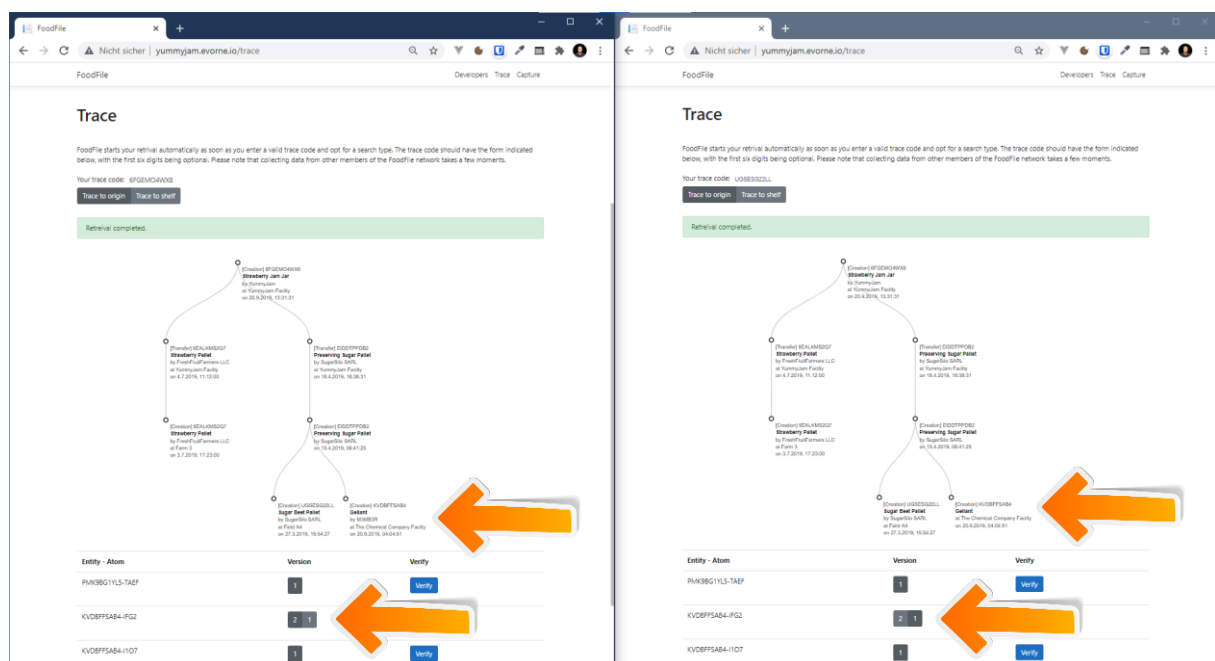


Figure 41: 'Histography Visualization' scenario result

### 8.4.18 Closing notes on resilient histography

The previous sections featured the support count as common indicator for information validity. This is a reasonable approach as the measure represents how many FoodFile members decided to adopt a certain information. In a more advanced implementation, the signature list could serve as an additional important factor for a validity metrics.

## 8.5 Comparison against Ruiz-Garcia et al.

In 2008, Ruiz-Garcia et al. published “[a] model and prototype implementation for tracking and tracing agricultural batch products along the food chain”. Their work resembles and potentially underpins the approach taken in the herein presented concept.

### 8.5.1 Similarities

Ruiz-Garcia et al. base their work on the finding that companies are legally required to store their supply-chain partners one step backwards and one step forward. The idea is to not enforce any data maintenance exceeding this threshold. A Service Oriented Architecture is introduced where each supply chain participant hosts its own data on a respective server. The services communicate with each other through a standardized interface they expose to the web. A central server registry enables discovery and URL retrieval of other participants. Every node (service) is capable to initiate a trace. During execution, a breath first search along the supply chain is performed with the results being visualized in a tree chart. Up- and downstream traces are possible.

### 8.5.2 Differences

Ruiz-Garcia et al. present a more detailed support for storing metadata (temperatures), process information and geolocations. To handle this complexity, they introduce process IDs and distinguish between horizontal queries (the trace) and vertical queries (the process interlinkage). To resemble this functionality in FoodFile, it would be sufficient to introduce a new information type for ‘processes’ to offer the desired fields. Introducing new information types in FoodFile does not require algorithmic adjustments.

FoodFile participants are free to choose the amount of information they wish to store on their instance. It can be what is required by law or include knowledge imported from other members. FoodFile incorporates the required tools

- to handle the resulting redundancy in the trace algorithm.
- to benefit from the resulting redundancy in case of members going offline (power outages, bankruptcy, etc.) and to maintain resilience regarding audit trails.

Ruiz-Garcia et al. do not mention any data auditing, verification mechanism, or data access management. While Ruiz-Garcia et al. logically separate information on the level of batches and supply chain steps, FoodFile additionally segregates data on the type of information. This approach allows for very granular auditing, verification and access management, all of which are conceptually covered and tested.

### 8.5.3 Conclusion

The conceptual commonalities of the compared approaches confirm that decentralized data maintenance is a desirable aspect in food supply chains. FoodFile has a similar approach to

trace across multiple nodes than what Ruiz-Garcia et al. have found to be an appropriate and working solution.

FoodFile extends the proposed architecture of Ruiz-Garcia et al. by taking inspiration from the DLT domain. As a result, it offers additional features such as redundancy, auditing, verification and access management.

## 9 Conclusion

This work has presented a new concept for a system that enables bidirectional tracing of individual food items across the entire supply chain; while maintaining the required data in a decentralized fashion on the respective production sites.

Food supply chain traceability is of public interest due to four major reasons:

- It improves public health by decreasing response times during foodborne outbreaks.
- It is expected to increase customer confidence by making detailed product information accessible and reduce food fraud.
- It reduces food waste by allowing for precisely targeted recalls and by enabling efficient retailing, e.g. applying a First-Expire-First-Out strategy.
- It offers potential for cost savings during production, as the knowledge of product flow and supply chain visibility can improve inventory management and demand forecasting.

These public interests manifest in legal requirements worldwide. Today, companies in the EU and US are obliged to maintain records about their suppliers and customers. In 2015, China has set the objective to establish a whole process food tracing system. Notwithstanding the desirability and the legal foundation, the common practice of recordkeeping today is shaped by manual work. Case studies have shown that multiple days of human investigation are needed to conduct a successful trace from a retailer back to the production origin(s).

The conceptualization of food tracing systems has been subject to scientific research and industry efforts. Yet, a commonly accepted practice or implementation does not exist. Recent developments indicate a trend towards Blockchain- or DLT-based approaches. As a preparing step for the primary contribution of this work, which is the presented concept 'FoodFile', the suitability of these technologies for the given problem domain has been analyzed. The major finding of this analysis is that the added complexity implied by Blockchain / DLT is not justified by the potential advantages the technologies generally offer, as these advantages apply only partially in the domain of food supply chain tracing. For a more detailed summary, see the conclusion of the suitability analysis of DLTs for food tracing systems: 5.3.6.

The presented concept was built to deliver the core functionality – bidirectional whole supply chain tracing for individual food entities – combined with privacy and trust related objectives (5.1). The privacy requirement of on-site data storage led to a decentralized design. To enable automated trace coordination in a decentralized system requires unified data exchange. FoodFile distinguishes six types (6.2) of information that can be tied to a physical entity in order to describe its flow through the supply chain. It is assumed that the respective supply chain members store their own process contribution in terms of the specific entity. As this naturally implies a fragmented knowledge distribution, the model introduces a recursive trace mechanism that can find and assemble information across all relevant supply chain steps on request.

Traces are directional: An upstream trace uses the next search iteration to follow up on the food-ingredients from the previous iteration. A downstream trace uses the next search iteration to follow up on the goods that were produced from the entities in the previous iteration.<sup>38</sup> Individual items are identified and labeled with a globally unique entity-ID. A trace is designed as a recursive breadth-first search that consecutively uncovers production steps and responsible supply chain actors. The search strategy guarantees that each supply chain step is queried only once per desired information. As replies are collected from multiple sources, they may be redundant or conflicting. The trace algorithm comprises automated merging and the possibility to compute statistics on conflicting information on request.

The concept introduces a granular access permission configuration: A supply chain member can allow or restrict data sharing individually for every logical combination of information type, entity and requesting supply chain member. If the access configuration allows for it, information can be made visible to consumers or other external stakeholders: The model is designed to optionally start the recursive trace leveraging a remote knowledge base instead of locally available information (which external stakeholders do not have). Multiple ways to distribute and copy information among supply chain actors to the degree it is desired by the network participants exist. Redundancy, a versioning system for the six information types and the possibility to digitally sign every individual piece of data facilitate a verifiable audit trail for each entity.

An MVP implementation was developed to serve a simulation-based evaluation. The simulation is composed of multiple scenarios, each of which is defined by three characteristics: First, a definition of an initial knowledge distribution among an exemplary supply chain, second, an appropriate trigger for the feature subject to validation and third, an expected result. 17 scenarios cover the different aspects of the objectives defined previously. As all the acceptance criteria was met when executing the simulation on the MVP implementation, it can be concluded that the proposed concept is valid.

This work contributes to the current state of research by introducing a practical solution for food tracing. The most important advantages are an increased trace speed and accuracy compared to manual investigations while respecting business interests of supply chain participants.

---

<sup>38</sup> Simplified. The model provided in chapter 6 has no bidirectional mapping between recursion depths and supply chain steps. Unrelated to when a supply chain member is found to be relevant during the trace process, this supply chain member will be queried about *all* entities the algorithm has discovered so far, no matter the current recursion depth.

## 9.1 Outlook

The logical next steps are an in-depth requirement engineering and a field test with food producers of different sizes and subsectors. The model in place today should undergo an additional verification to ensure it matches the user's needs. Many potential feature candidates must be evaluated, e.g. support for EPCIS, metadata storage (temperature, humidity, detailed production steps, etc.) and an improved location identification system capable to map facilities to members.



## 10 References

- [1] “Kontaminierte Eier in zwölf europäischen Ländern,” *ZEIT Online*, 10 Aug., 2017. <https://www.zeit.de/gesellschaft/zeitgeschehen/2017-08/fipronil-eier-europa-daenemark-rumaenien> (accessed: Nov. 22 2019).
- [2] “Was Sie über mit Fipronil belastete Eier wissen sollten,” *ZEIT Online*, 03 Aug., 2017. <https://www.zeit.de/wissen/umwelt/2017-08/fipronil-gift-eier-verbraucherschutz-bioeier-rueckruf-landwirtschaft> (accessed: Nov. 22 2019).
- [3] M. Radunski, “Woher stammen die Fipronil-Eier?,” *Frankfurter Allgemeine Zeitung*, 12 Jun., 2018. <https://www.faz.net/aktuell/wirtschaft/mehr-wirtschaft/fipronil-eier-skandal-die-ursachen-und-moegliche-folgen-15635382> (accessed: Nov. 21 2019).
- [4] Renate Künast, *Wieder #Fipronil in Eiern*. [Online]. Available: <https://twitter.com/RenateKuenast/status/1006499044670177283> (accessed: Nov. 22 2019).
- [5] Food and Agriculture Organization of the United Nations, *Food Traceability Guidance*. [Online]. Available: <http://www.fao.org/3/a-i7665e.pdf> (accessed: Nov. 21 2019).
- [6] R. Badia-Melis, P. Mishra, and L. Ruiz-García, “Food traceability: New trends and recent advances. A review,” *Food Control*, vol. 57, pp. 393–401, 2015, doi: 10.1016/j.foodcont.2015.05.005.
- [7] *Regulation (EC) No 178/2002*, 2002. Accessed: Nov. 22 2019. [Online]. Available: <https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32002R0178>
- [8] L. Zach, “Legal Requirements and Regulation for Food Traceability in the United States,” in *Advances in Food Traceability Techniques and Technologies*: Elsevier, 2016, pp. 237–260.
- [9] *Food Safety Law of the People's Republic of China*, 2015. [Online]. Available: [https://apps.fas.usda.gov/newgainapi/api/report/downloadreportbyfilename?filename=Amended%20Food%20Safety%20Law%20of%20China\\_Beijing\\_China%20-%20Peoples%20Republic%20of\\_5-18-2015.pdf\\_Beijing\\_China%2520-%2520Peoples%2520Republic%2520of\\_5-18-2015.pdf&usg=AOvVaw1v-Qtd1\\_mM87CNa14CKS2t](https://apps.fas.usda.gov/newgainapi/api/report/downloadreportbyfilename?filename=Amended%20Food%20Safety%20Law%20of%20China_Beijing_China%20-%20Peoples%20Republic%20of_5-18-2015.pdf_Beijing_China%2520-%2520Peoples%2520Republic%2520of_5-18-2015.pdf&usg=AOvVaw1v-Qtd1_mM87CNa14CKS2t)
- [10] *How Walmart brought unprecedented transparency to the food supply chain with Hyperledger Fabric: Case Study*. [Online]. Available: <https://www.hyperledger.org/resources/publications/walmart-case-study> (accessed: Nov. 24 2019).
- [11] A. W. Kennedy and J. McEntire, “Connecting the Dots with Whole Chain Traceability,” in *Food Traceability*, J. McEntire and A. W. Kennedy, Eds., Cham: Springer International Publishing, 2019, pp. 181–192.
- [12] Christine Leong, Tal Viskin, Robyn Stewart, *Tracing the supply chain: Research Report*. [Online]. Available: <https://www.accenture.com/us-en/insights/blockchain/food-traceability> (accessed: Nov. 24 2019).

- [13] IBM, *IBM Blockchain Website: Blockchain in Retail*. [Online]. Available: <https://www.ibm.com/blockchain/industries/retail> (accessed: Nov. 24 2019).
- [14] IBM, *About IBM FoodTrust*. [Online]. Available: <https://www.ibm.com/downloads/cas/8QABQBDR> (accessed: Nov. 18 2019).
- [15] 21 Authors (IBM), *Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains*. [Online]. Available: <https://arxiv.org/pdf/1801.10228.pdf> (accessed: Oct. 30 2019).
- [16] *United States public law 107-188 / Public Health Security And Bioterrorism Preparedness and Response Act of 2002*.
- [17] *Report to Congress On Enhancing Tracking and Tracing of Food and Recordkeeping*, 2016.
- [18] J. Guzewich and B. Miller, "Public Health," in *Food Traceability*, J. McEntire and A. W. Kennedy, Eds., Cham: Springer International Publishing, 2019, pp. 27–49.
- [19] M. Southall, "Industry Benefits," in *Food Traceability*, J. McEntire and A. W. Kennedy, Eds., Cham: Springer International Publishing, 2019, pp. 51–62.
- [20] B. Blakistone and S. Mavity, "Seafood," in *Food Traceability*, J. McEntire and A. W. Kennedy, Eds., Cham: Springer International Publishing, 2019, pp. 97–111.
- [21] L. J. Lupo, "A Chain of Linked Nuances," in *Food Traceability*, J. McEntire and A. W. Kennedy, Eds., Cham: Springer International Publishing, 2019, pp. 113–132.
- [22] T. Bhatt, G. Buckley, J. C. McEntire, P. Lothian, B. Sterling, and C. Hickey, "Making traceability work across the entire food supply chain," *Journal of food science*, 78 Suppl 2, B21-7, 2013, doi: 10.1111/1750-3841.12278.
- [23] R. Jedermann, J. P. Edmond, and W. Lang, "Shelf life prediction by intelligent RFID," *Dynamics in logistics - First international conference, LDIC 2007 Bremen, Germany*, pp. 231–238, 2007.
- [24] *ISO/IEC 18004:2015*.
- [25] *ISO/IEC 16022:2006-09*.
- [26] *EPC™ Radio-Frequency Identity Protocols Generation-2 UHF RFID*.
- [27] T. Bhatt, C. Hickey, and J. C. McEntire, "Pilot projects for improving product tracing along the food supply system," *Journal of food science*, 78 Suppl 2, B34-9, 2013, doi: 10.1111/1750-3841.12298.
- [28] S. Haber and W.S. Stornetta, "How to time-stamp a digital document," *J. Cryptology*, vol. 3, no. 2, 1991, doi: 10.1007/BF00196791.
- [29] S. Konst, *Sichere Log-Dateien auf Grundlage kryptographisch verketteter Einträge*: Universitätsbibliothek Braunschweig, 2000.

- [30] Satoshi Nakamoto, *Bitcoin: A peer-to-peer electronic cash system*,” <http://bitcoin.org/bitcoin.pdf>.
- [31] L. Lamport, R. Shostak, and M. Pease, “The Byzantine Generals Problem,” *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982, doi: 10.1145/357172.357176.
- [32] M. Castro and B. Liskov, “Practical byzantine fault tolerance and proactive recovery,” *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, 2002, doi: 10.1145/571637.571640.
- [33] P. Treleaven, R. Gendal Brown, and D. Yang, “Blockchain Technology in Finance,” *Computer*, vol. 50, no. 9, pp. 14–17, 2017, doi: 10.1109/MC.2017.3571047.
- [34] D. Mills *et al.*, “Distributed Ledger Technology in Payments, Clearing, and Settlement,” *FEDS*, vol. 2016, no. 095, 2016, doi: 10.17016/FEDS.2016.095.
- [35] F. M. Bencic and I. Podnar Zarko, “Distributed Ledger Technology: Blockchain Compared to Directed Acyclic Graph,” in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, Vienna, Jul. 2018 - Jul. 2018, pp. 1569–1570.
- [36] A. Narayanan and J. Clark, “Bitcoin's academic pedigree,” *Commun. ACM*, vol. 60, no. 12, pp. 36–45, 2017, doi: 10.1145/3132259.
- [37] Mike Hearn, *Corda: A distributed ledger: Version 0.5*.
- [38] R. Maull, P. Godsiff, C. Mulligan, A. Brown, and B. Kewell, “Distributed ledger technology: Applications and implications,” *Strategic Change*, vol. 26, no. 5, pp. 481–489, 2017, doi: 10.1002/jsc.2148.
- [39] A. Pinna and W. Ruttenberg, *Distributed ledger technologies in securities post-trading: Revolution or evolution?* Frankfurt am Main, Germany: European Central Bank, 2016. [Online]. Available: <http://hdl.handle.net/10419/154625>
- [40] L. W. Cong and Z. He, “Blockchain Disruption and Smart Contracts,” *The Review of Financial Studies*, vol. 32, no. 5, pp. 1754–1797, 2019, doi: 10.1093/rfs/hhz007.
- [41] P. Tasca and T. Thanabalasingham, “Ontology of Blockchain Technologies. Principles of Identification and Classification,” *SSRN Journal*, 2017, doi: 10.2139/ssrn.2977811.
- [42] B. W. Lampson, “How to build a highly available system using consensus,” in *Lecture Notes in Computer Science, Distributed Algorithms*, G. Goos, J. Hartmanis, J. Leeuwen, Ö. Babaoğlu, and K. Marzullo, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 1–17.
- [43] L. Lamport, “The part-time parliament,” *ACM Trans. Comput. Syst.*, vol. 16, no. 2, pp. 133–169, 1998, doi: 10.1145/279227.279229.
- [44] D. Ongaro and J. Ousterhout, “In Search of an Understandable Consensus Algorithm,” in *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, 2014, pp. 305–320.

- [45] Q. He, N. Guan, M. Lv, and W. Yi, "On the Consensus Mechanisms of Blockchain/DLT for Internet of Things," in *2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES)*, 2018, pp. 1–10.
- [46] J. Kwon, "Tendermint : Consensus without Mining," in 2014.
- [47] J. Kwon and E. Buchmann, *Cosmos: A Network of Distributed Ledgers*. [Online]. Available: <https://cosmos.network/cosmos-whitepaper.pdf> (accessed: Feb. 13 2020).
- [48] S. de Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "Pbft Vs Proof-Of-Authority: Applying The Cap Theorem To Permissioned Blockchain," 2017.
- [49] Leemon Baird, Mance Harmon, Paul Madsen, *Hedera: A public Hashgraph Network & Governing Council: Whitepaper V.2.0*.
- [50] Leemon Baird, *The Swirlds Hashgraph Consensus Algorithm: Fair, Fast Byzantine Fault Tolerance*.
- [51] Serguei Popov, *The Tangle: Version 1.4.3*.
- [52] GS1, *EPC Information Services (EPCIS) Standard*.
- [53] Optel Group, *Optel Group Website*. [Online]. Available: <https://www.optelgroup.com/> (accessed: Mar. 27 2020).
- [54] IBM, *IBM FoodTrust Developer-Zone*. [Online]. Available: <https://github.com/IBM/IFT-Developer-Zone> (accessed: Mar. 27 2020).
- [55] Ambrosus, *Ambrosus White Paper: Version 8.1* (accessed: Mar. 27 2020).
- [56] "Traceability (Product Tracing) in Food Systems: An IFT Report Submitted to the FDA, Volume 1: Technical Aspects and Recommendations," *Comprehensive Reviews in Food Science and Food Safety*, vol. 9, no. 1, pp. 92–158, 2010, doi: 10.1111/j.1541-4337.2009.00097.x.
- [57] D. Mao, Z. Hao, F. Wang, and H. Li, "Innovative Blockchain-Based Approach for Sustainable and Credible Environment in Food Trade: A Case Study in Shandong Province, China," *Sustainability*, vol. 10, no. 9, p. 3149, 2018, doi: 10.3390/su10093149.
- [58] M. Thakur and C. R. Hurburgh, "Framework for implementing traceability system in the bulk grain supply chain," *Journal of Food Engineering*, vol. 95, no. 4, pp. 617–626, 2009, doi: 10.1016/j.jfoodeng.2009.06.028.
- [59] T. Chen, K. Ding, S. Hao, G. Li, and J. Qu, "Batch-based traceability for pork: A mobile solution with 2D barcode technology," *Food Control*, vol. 107, p. 106770, 2020, doi: 10.1016/j.foodcont.2019.106770.
- [60] W. He, N. Zhang, P. S. Tan, E. W. Lee, T. Y. Li, and T. L. Lim, "A secure RFID-based track and trace solution in supply chains," in *2008 6th IEEE International Conference on Industrial Informatics*, Daejeon, South Korea, Jul. 2008 - Jul. 2008, pp. 1364–1369.

- [61] W. He, E. L. Tan, E. W. Lee, and T. Y. Li, "A solution for integrated track and trace in supply chain based on RFID & GPS," in *2009 IEEE Conference on Emerging Technologies & Factory Automation*, Palma de Mallorca, Spain, Sep. 2009 - Sep. 2009, pp. 1–6.
- [62] Z. Li, G. Liu, L. Liu, X. Lai, and G. Xu, "IoT-based tracking and tracing platform for pre-packaged food supply chain," *Industr Mngmnt & Data Systems*, vol. 117, no. 9, pp. 1906–1916, 2017, doi: 10.1108/IMDS-11-2016-0489.
- [63] L. Ruiz-Garcia, G. Steinberger, and M. Rothmund, "A model and prototype implementation for tracking and tracing agricultural batch products along the food chain," *Food Control*, vol. 21, no. 2, pp. 112–121, 2010, doi: 10.1016/j.foodcont.2008.12.003.
- [64] F. Tian, "A supply chain traceability system for food safety based on HACCP, blockchain & Internet of things," in *2017 International Conference on Service Systems and Service Management*, Dalian, China, Jun. 2017 - Jun. 2017, pp. 1–6.
- [65] Trent McConaghy, Rodolphe Marques, Andreas Müller, Dimitri De Jonghe, Troy McConaghy, Greg McMullen, Ryan Henderson, Sylvain Bellemare, Alberto Granzotto, *BigchainDB: A Scalable Blockchain Database*.
- [66] BigchainDB GmbH, *BigchainDB 2.0: The Blockchain Database*. [Online]. Available: <https://www.bigchaindb.com/whitepaper/>
- [67] O. Bermeo-Almeida, M. Cardenas-Rodriguez, T. Samaniego-Cobo, E. Ferruzola-Gómez, R. Cabezas-Cabezas, and W. Bazán-Vera, "Blockchain in Agriculture: A Systematic Literature Review," in *Communications in Computer and Information Science, Technologies and Innovation*, R. Valencia-García, G. Alcaraz-Mármol, J. Del Cioppo-Morstadt, N. Vera-Lucio, and M. Bucaram-Leverone, Eds., Cham: Springer International Publishing, 2018, pp. 44–56.
- [68] T. Burke, "Blockchain in Food Traceability," in *Food Traceability*, J. McEntire and A. W. Kennedy, Eds., Cham: Springer International Publishing, 2019, pp. 133–143.
- [69] M. Rauchs *et al.*, "Distributed Ledger Technology Systems: A Conceptual Framework," *SSRN Journal*, 2018, doi: 10.2139/ssrn.3230013.
- [70] *Hyperledger Fabric: Release 2.0 Docs*. Key Concepts. [Online]. Available: [https://hyperledger-fabric.readthedocs.io/en/release-2.0/key\\_concepts.html](https://hyperledger-fabric.readthedocs.io/en/release-2.0/key_concepts.html) (accessed: Apr. 13 2020).
- [71] *Hyperledger Fabric: Open, Proven, Enterprise-grade DLT*. [Online]. Available: <https://www.hyperledger.org/projects/fabric> (accessed: Apr. 13 2020).
- [72] Dr. Gavin Wood, *Ethereum: A Secure Decentralised Generalised Transaction Ledger*.
- [73] Niclas Kannengießer, Sebastian Lins, Tobias Dehling, Ali Sunyaev, *Mind the Gap: Trade-Offs Between Distributed Ledger Technology Characteristics: Working Paper* (accessed: May 10 2020).

- [74] Elastic NV, *Elastic Stack and Product Documentation*. [Online]. Available: <https://www.elastic.co/guide/index.html> (accessed: May 24 2020).
- [75] Elastic NV, *Elasticsearch GitHub Repository & Readme*. [Online]. Available: <https://github.com/elastic/elasticsearch> (accessed: Apr. 24 2020).
- [76] Apache Lucene 7.4.0 Documentation. [Online]. Available: [https://lucene.apache.org/core/7\\_4\\_0/index.html](https://lucene.apache.org/core/7_4_0/index.html) (accessed: May 24 2020).
- [77] Microsoft, *F# Documentation*. [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/fsharp/> (accessed: Apr. 24 2020).
- [78] *F# GitHub Repository*. [Online]. Available: <https://github.com/fsharp/fsharp> (accessed: May 24 2020).
- [79] Facebook Inc., *React Documentation*. [Online]. Available: <https://reactjs.org/docs/> (accessed: May 24 2020).
- [80] Facebook Inc., *React GitHub Repository*. [Online]. Available: <https://github.com/facebook/react/> (accessed: May 24 2020).