Personal and Ubiquitous Computing
https://doi.org/10.1007/s00779-020-01417-z

ORIGINAL ARTICLE

A secure and extensible blockchain-based data provenance framework for the Internet of Things

Marten Sigwart[1] · Michael Borkowski[2] · Marco Peise[3] · Stefan Schulte[1] · Stefan Tai[3]

Received: 4 November 2019 / Accepted: 18 May 2020

Abstract

As data collected and provided by Internet of Things (IoT) devices power an ever-growing number of applications and services, it is crucial that this data can be trusted. Data provenance solutions combined with blockchain technology are one way to make data more trustworthy by providing tamper-proof information about the origin and history of data records. However, current blockchain-based solutions for data provenance fail to take the heterogeneous nature of IoT applications and their data into account. In this work, we identify functional and non-functional requirements for a secure and extensible IoT data provenance framework, and conceptualise the framework as a layered architecture. Evaluating the framework using a proof-of-concept implementation based on Ethereum smart contracts, we conclude that our framework can be used to realise data provenance concepts for a wide range of IoT use cases. While blockchain technology generally poses constraints on scalability and privacy, we discuss multiple solutions aiming to overcome these issues.

Keywords Internet of things · IoT · Data provenance · Blockchains · Smart contracts

1 Introduction

The Internet of Things (IoT) is transforming many areas of our everyday lives. IoT technologies such as GPS, RFID-based identification, sensors, and low-resource computing platforms like the Raspberry Pi already play important roles in various domains, e.g. mobility, logistics, healthcare, and retail [1]. Since data collected by these technologies find application in an increasing number of use cases, ensuring the trustworthiness of such data is of high importance [2].

Data provenance systems are one way to ensure trustworthiness in the IoT [3]. These systems can provide information about the origin and evolution of data such as the various stages of data creation and data modifications, who initiated them, and when and how they took place [4]. In short, a data provenance system tracks who has created, updated, deleted, and in some cases read particular data points [4]. In order to trust the information provided by a provenance system, it is essential that the provenance system stores information in a tamper-proof and replicable way [5].

Traditionally, in distributed settings, participants must either trust each other or an independent third party to store data in a tamper-proof way. With the advent of blockchain technologies, the requirement of such trust in a central authority is eliminated. Instead, a decentralised network can be established, acting as a distributed, tamper-proof ledger [6]. Thus, leveraging blockchain technology in a data provenance solution for the IoT is a promising choice [6].

Since the IoT is characterised by a multitude of use cases and potential application areas [1], an IoT data provenance solution should account for this diversity [7]. However, approaches aiming at scenario-agnostic blockchain-based data provenance solutions for the IoT offer so far no concrete software solutions [8, 9], while more concrete

✉ Marten Sigwart
m.sigwart@dsg.tuwien.ac.at

Michael Borkowski
michael.borkowski@dlr.de

Marco Peise
mp@ise.tu-berlin.de

Stefan Schulte
s.schulte@dsg.tuwien.ac.at

Stefan Tai
st@ise.tu-berlin.de

1 TU Wien, Vienna, Austria

2 Institute of Flight Guidance, German Aerospace Center (DLR), Braunschweig, Germany

3 TU Berlin, Berlin, Germany

Published online: 16 June 2020

solutions to data provenance in the IoT focus on specific application areas, for instance, supply chains [10–12], health monitoring systems [13], or digital forensics [14].

Hence, we propose a generic blockchain-based data provenance framework for the IoT. Hereby, "generic" refers to the fact that the framework should not be limited to be applicable within one particular use case scenario, but rather should facilitate the recording of provenance data in a tamper-proof way in arbitrary IoT scenarios. The advantages of a generic framework are the easier adoption and faster implementation of provenance concepts by new use cases as well as the interoperability of applications that use the framework.

This paper extends our previous work [15] (i) by providing an additional motivational use case scenario from digital forensics, (ii) by extending the evaluation by taking the additional use case into account as well as by conducting a more rigorous evaluation of transaction throughput and latency of the solution, and (iii) by providing a discussion about possible solutions to the scalability and privacy constraints of a blockchain-based data provenance solution. Furthermore, the related work has been updated and extended. The contributions of this work are as follows:

- We define functional and non-functional requirements for a generic IoT data provenance framework.
- We conceptualise and implement an IoT data provenance framework consisting of smart contracts using a generic data model to provide provenance functionality for a wide range of IoT use cases.
- We present an evaluation of the framework with regard to the defined requirements using a proof-of-concept implementation with Ethereum smart contracts.
- We discuss possible solutions to the scalability and privacy limitations of blockchain-based applications for the IoT.

In the following, we briefly introduce the underlying concepts of data provenance and further motivate our work by discussing exemplary use case scenarios (Section 2), define the requirements for a generic blockchain-based data provenance framework (Section 3), and explain the concepts, architecture, and proof-of-concept implementation of our solution (Section 4). In Section 5, we evaluate the framework with regard to the defined requirements. The results of the evaluation are discussed in Section 6. Section 7 provides an overview of the related work. Finally, Section 8 concludes the paper.

2 Background and motivation

Data provenance, sometimes also known as lineage or pedigree [16], identifies the derivation history of data [4]. While originally used for works of art, data provenance is now relevant in a wide range of use cases, since data provenance mechanisms help establish a certain level of trust in data by providing information about its creation, access, and transfer [4]. Provided provenance data is secured, forgery, alteration, or repudiation of data can be prevented [5].

Accordingly, data provenance solutions can help establish trust in data in the IoT [2]. In the following, we describe three exemplary use cases for data provenance in the IoT.

2.1 Vaccine supply chains

Immunisation programmes depend on functional, end-to-end supply chains [17]. In all phases of the supply chain—from procurement to last-mile distribution—it is of significant importance that vaccines remain in a temperature range of around 2–8 °C. Otherwise, vaccines lose their effectiveness. An unbroken, temperature-controlled link from producer to consumer is also referred to as the cold chain [17]. Figure 1 shows a simplified model of such a cold chain enhanced with IoT-technologies, e.g., RFID, GPS, and sensors [1]. These technologies continuously track temperature, location, and other conditions of the vaccines. This provenance data helps establish confidence in the quality of vaccines and exposes any weak links along the cold chain. As mentioned in Section 1, an important requirement for provenance systems is that recorded provenance information is replicable and tamper-proof. Recent scandals of vaccine counterfeiting (e.g., [18]) have confirmed the urgency of these requirements in vaccine supply chain systems. Hence, an IoT-powered data provenance system based on blockchain technology could provide benefits in vaccine supply chain scenarios. In particular, it enables the backtracking of errors in case of breakage of the cold chain, and it helps build and keep trust in immunisation programmes by preventing counterfeit vaccines from entering the supply chain.

2.2 Health monitoring systems

Health monitoring scenarios are another application area where secure data provenance records can provide additional trust in data [13]. In such scenarios, sensors monitor health conditions of patients such as the patient's heart rate and blood pressure. Combined with a real-time analytics service, the sensor readings can be used to notify relatives and health professionals in case of medical emergencies, such as a heart attack. Ideally, the root cause of such a notification is verifiable, i.e. it is possible to trace back a notification to the individual sensor readings which caused it. Otherwise, if the circumstances that led to the emergency notification are unclear, the possibility that the notification

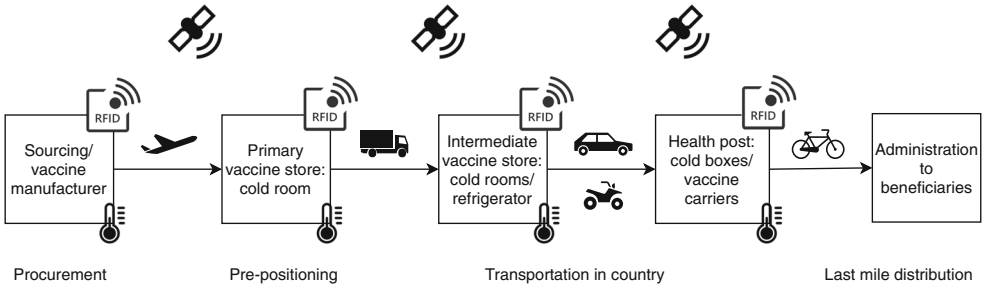

Fig. 1 IoT-enhanced vaccine supply chain

was caused by a malicious attack tampering with the system cannot be excluded. Having secure provenance data for a notification, such as information about the individual sensor readings and analytics involved, ensures the reliability and accuracy of the system.

2.3 Digital forensics

A further use case where data provenance records secured via blockchain technology can provide benefits is digital forensics, for instance in scenarios involving accidents of autonomous vehicles [14]. Ideally, in such cases, provenance data is available that uncovers the true cause of the accident. At the same time, it should be impossible for anyone to illegally alter the provenance data after an accident occurred. Consider the example in Fig. 2. The provenance data may consist of data provided by the manufacturer from when the car was manufactured, maintenance data provided by garages and workshops, and data collected while driving by various sensors within and around the car such as speed, road conditions, and outside temperature. This data is submitted via transactions to the provenance framework on the blockchain. When an accident occurs, the car owner submits a liability claim to the insurance company. The insurance company can query the provenance framework and depending on the evidence forward the liability claim to manufacturers or garages. The blockchain ensures the integrity of the data and that the source of the data can be trusted. As such, it can help investigators resolve any disputes of liability claims.

3 Requirements

This section defines the functional and non-functional requirements of a generic data provenance framework for the IoT. Such a framework needs to record provenance data for addressable data points, i.e. data that has a unique ID [7]. In accordance with [4] and [7], we derive the functional requirements (Reqs. (1)–(7)) of our framework. Furthermore, in accordance with Hasan et al. [5], we define non-functional requirements that need to be fulfilled by a tamper-proof data provenance system for the IoT (Reqs. (8)–(11)).

1. *Provenance abstraction:* The framework needs to provide generic data provenance capturing, storing, and querying functionality which can be adopted by provenance use cases to map their specific requirements.

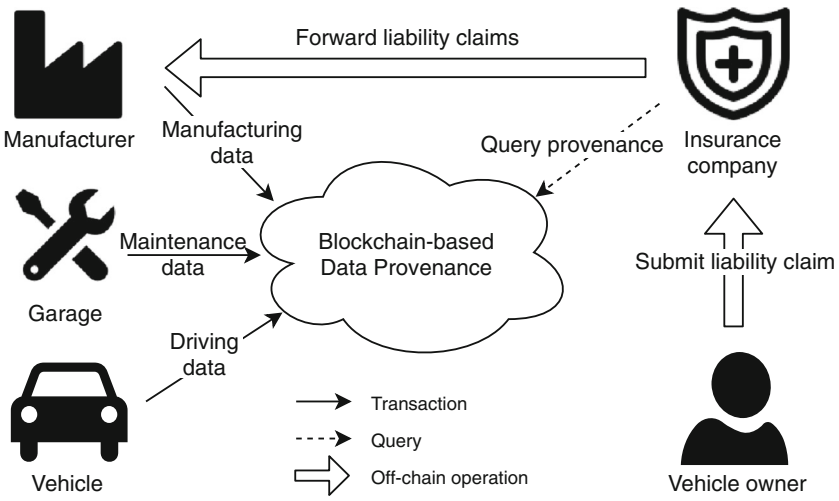

Fig. 2 An IoT data provenance framework secured via blockchain technology could resolve disputes of liability claims

2. *High-level and low-level provenance:* Provenance records represent high-level as well as low-level data points. Low-level data points stem from low-level devices such as sensors. High-level data points do not have a single physical origin (such as a sensor reading) but represent more abstract concepts, e.g. some physical object in a supply chain or an analytics result based on multiple inputs.
3. *Completeness:* The provenance records of a data point can be regarded complete if every relevant action which has ever been performed on a data point is gathered [5]. Here, relevance implies that some actions can be neglected if they do not contribute to the provenance data of a particular data point.
4. *Creation of lineage:* Provenance records for a data point can be created based on the last provenance record for that same data point. This enables the creation of lineage. For instance, the lineage of a data point representing a physical good travelling along a supply chain can be tracked by creating a new provenance record based on the old one at each critical step of the supply chain.
5. *Derivation:* A provenance record entails references to the provenance records of the data points that led to its creation. For instance, a provenance record for an analytics result based on value readings from multiple sensors must not only contain information about the sensor values but must also be able to access provenance data of those same values, such as location and time of recording.
6. *Provenance for modifications of data points:* The framework enables the tracking of the modification history of a specific data point. For instance, if an analytics result runs through multiple stages of different calculations, the history of these calculations can be tracked.
7. *Parallel provenance:* Multiple provenance records for the same data point can exist in parallel, e.g. one provenance trace might track the ownership of a data point (e.g. the current owner of a physical good), while another provenance record is tracking the location of that same data point.
8. *Integrity:* Integrity mandates that provenance records cannot be manipulated or modified by an adversary in any way. This is crucial for establishing trust in the data. Without guaranteed integrity, clients can potentially repudiate provenance records.
9. *Availability and fault tolerance:* The provenance framework needs to enable clients to reliably reconstruct the creation and modification history of data. For that, the provenance data needs to be available when requested by clients even in the case that some parts of the system fail.
10. *Privacy:* In general, privacy requires that the unauthorised collection, storage, and access of sensitive data is prevented. Privacy encompasses confidentiality and anonymity. On the one hand, provenance records of IoT devices may contain sensitive data, e.g. in a health monitoring system. It is therefore vital to keep this data confidential, i.e. access to this data from unauthorised entities needs to be prevented. On the other hand, even when the data itself is kept confidential, malicious actors might still be able to create user profiles by identifying user-specific patterns. As such, traceability of provenance data needs to be prevented to ensure the anonymity of users. In the following, we refer to both notions simply as privacy.
11. *Scalability:* A provenance system is required to have a reasonable expenditure. Storing and accessing provenance data must have a low overhead. Especially, even though some IoT devices are resource-constrained, they must not be excluded from participating in the provenance framework. Furthermore, applications in the IoT potentially deal with massive amounts of data and very frequent data updates which a provenance solution needs to account for.

4 IoT data provenance framework

This section presents the proposed generic blockchain-based data provenance framework for the IoT. As mentioned in Section 1, major benefits of a generic framework are interoperability between applications using the framework and the facilitated implementation of provenance concepts for new IoT use cases. Figure 3 displays the core architecture of the framework. It consists of three layers all embedded within a blockchain smart contract platform. Each layer represents a different level of abstraction with a diverse set of responsibilities within the framework: The storage layer is primarily concerned with low-level representation and storage of provenance data, and the generic provenance layer provides general-purpose provenance functionality, while the specific provenance layer can be modified by use cases to fine-tune the framework to their specific requirements. To this end, Section 4.1 describes the data model used for the representation of provenance data within the framework, and Section 4.2 explains the storage layer, whereas Section 4.3 and Section 4.4 describe the generic and specific provenance layers, respectively.

We provide a proof-of-concept implementation of the framework consisting of smart contracts written in Solidity—a smart contract language for the Ethereum Virtual Machine (EVM). As such, the prototype can

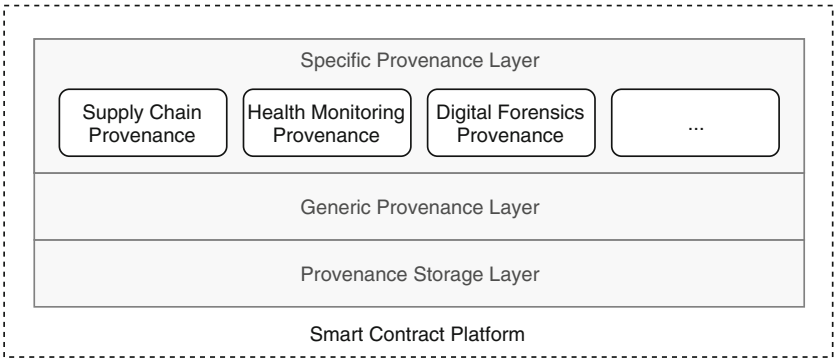

Fig. 3 Data provenance framework architecture

be deployed on any EVM-compatible blockchain, e.g. public permissionless blockchains such as Ethereum and Ethereum Classic, or private permissioned blockchains such as Hyperledger Burrow [19]. However, the framework is not restricted to EVM-based blockchains. The described concepts could also be implemented on any blockchain platform that provides sufficient scripting capabilities. The prototype is available as open-source software on Github.[1]

4.1 Data model

Provenance data varies depending on the specific domain or application [7]. Since the IoT domain is characterised by heterogeneous applications and a wide range of possible use cases [1], there is a need for a generalised provenance data model suitable for IoT applications. As the underlying data model, our framework utilises a data provenance model specifically designed for the IoT by Olufowobi et al. [7]. Instead of taking a document-centric view on provenance data such as models like the Open Provenance Model (OPM) [20] or the PROV data model [21] which track agents performing actions on documents, the model by Olufowobi et al. focuses on the infrastructure of agents like sensors, devices, analytics services, and the exchanged data. The authors argue that in IoT environments provenance can be recorded in terms of creations or modifications of data by agents. No distinction between agents and actions is necessary since a fine-grained definition of agents already explains the action performed on the data. The model can therefore account for fine-grained provenance data while keeping a low overhead.

The model defines a data point as a uniquely identifiable and addressable piece of data. In the context of IoT systems, these can be sensor readings, complex analytics results derived from sensor readings, actuator commands, etc. The function *addr*(*dp*) denotes the address/ID of a data point *dp*. The function *inputs*(*dp*) refers to the set of input data points that have contributed to the creation or modification of a data point.

The model defines a provenance event as the moment an IoT system reaches a specific state which requires the collection of provenance data for a particular data point. When a provenance event occurs, information of interest for provenance about the state of the IoT system need to be recorded. The function *context*(*dp*) denotes such information. The context varies depending on the IoT application. An example of how the context parameter might be structured is displayed in Fig. 4. Agents (e.g. sensors, devices, persons, organisations, etc.) create and/or modify data points. Here, agents are defined recursively, i.e. agents can contain other agents (such as devices containing several sensors). Information related to the specific provenance event, such as events that triggered the creation/modification of the data point are defined in the execution context. Furthermore, time and location information may be added to the context as well.

Finally, the model defines a provenance record as a tuple associating the address of a data point with the set of provenance records of its input data points and the specific context of the corresponding provenance event. The provenance function *prov*(*dp*), providing the provenance record for some data point *dp*, is defined as follows:

$$prov\ :\ dp\ \mapsto\ \langle addr\ (dp),\ \{prov\ (idp)|\ \forall idp\ \in\ inputs\ (dp)\},\ context\ (dp)\rangle$$

Note that this definition of provenance allows for the description of both creation and modification of data points. In the former case, the set of input provenance records contains an empty set. In the latter case, the set of input provenance records contains the provenance record of the data point before modification, i.e. $prov(dp') = \langle addr(dp'), \{prov(dp), ...\}, context(dp')\rangle$.

The described data provenance model combined with blockchain technology builds the basis for our IoT data provenance framework. This way, the framework can not only represent provenance data for various IoT use cases, but also provides integrity guarantees for the stored records.

[1] https://github.com/msigwart/iotprovenance

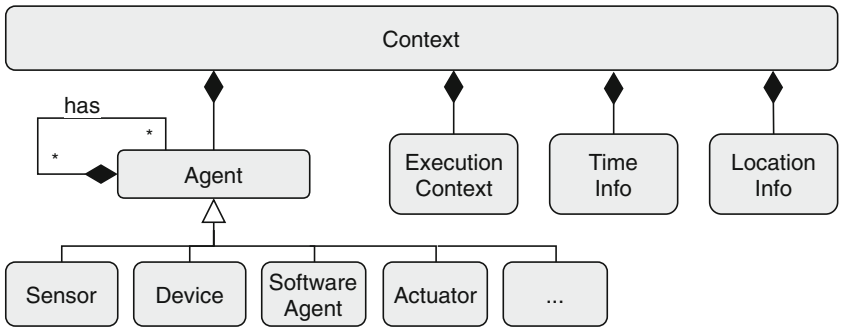

Fig. 4 UML diagram of an exemplary provenance record's context [7]

4.2 Storage layer

This layer is responsible for the low-level storage of provenance records. It contains the generic representation for provenance records as defined by the data provenance model, as well as basic functionality to create, retrieve, update, and delete provenance records. Delete refers to the invalidation of provenance records, since truly deleting data from a blockchain is not possible. An invalidated provenance record cannot be used as input for subsequent provenance records.

Listing 1 displays an excerpt of the smart contract implementing this layer. The internal representation of provenance records (Lines 2–7) closely resembles the model discussed above with the field *tokenId* representing the ID of a data point. However, not only are data points addressable, but also the provenance records themselves. Addressable provenance records allow the storage layer to manage provenance records as a mapping from provenance IDs to provenance records: $addr\,(prov) \mapsto prov$ where the function *addr*(*prov*) represents the ID of a provenance record *prov*. The mapping is implemented via the *mapping* keyword (Line 8).

The contract exposes an API for creating, retrieving, updating, and deleting (i.e. invalidating) provenance records. Note that, while functionality for retrieving provenance records is exposed publicly via the *public* keyword (Lines 10ff), functionality for creating, updating, and deleting records is protected via the *internal* keyword (Lines 13ff). These functions cannot be accessed publicly, but are accessible from inheriting contracts such as the contract representing the generic provenance layer. Read functions are publicly accessible since these functions do not alter the state of the contract.

```
1  contract Storage {
2  struct ProvenanceRecord {
3      uint tokenId;
4      uint[] inputProvenanceIds;
5      string context;
6      uint index;
7  }
8  mapping (uint => ProvenanceRecord) records;
9  uint[] provenanceIndex;
10 function getProvenance(uint provId)
11 public view returns (uint tokenId, ...) {...}
12 ...
13 function createProvenance(uint provId, ...)
14 internal returns (uint index) {...}
15 ...
16 }
```

Listing 1 Storage contract (excerpt)

4.3 Generic provenance layer

The generic provenance layer's main purpose is to provide general-purpose provenance functionality on top of the storage layer; i.e. it provides features that are universally applicable for a wide range of provenance use cases. For this, it defines the ownership of data points (Section 4.3.1) and how to associate provenance records with data points (Section 4.3.2).

4.3.1 Ownership of data points

While blockchain technology can guarantee the integrity of data provenance records once those records have entered the system, mechanisms need to be in place to ensure that the records that enter the system are correct. As a first step, we aim to prevent the creation of provenance records by arbitrary clients, i.e. if client A generates some data point dp_0, we want to make sure that only client A (or any client authorised by client A) is able to create provenance records for dp_0. Thus, we introduce the notion of ownership of data points: Each data point belongs to a specific client of the system, and only the owner or a client authorised by the owner can create provenance records for it. If an unauthorised client tries to create a provenance record for a data point, the system raises an error.

The notion of ownership is closely related to so-called tokens, i.e. smart contracts deployed on a blockchain that represent a kind of digital asset [22]. Our framework leverages tokens to introduce ownership of data points. Each data point is represented by a single token and a single token

identifies exactly one data point. This one-to-one mapping allows us to identify the owner of a data point by identifying the owner of a particular token.

Each token acts as an entry ticket to the provenance framework. To create provenance records for a particular data point, a client first has to become the token owner or be approved by the owner of the corresponding token. The prototype uses the Ethereum token standard ERC721[2] which defines a common interface for non-fungible assets, for instance functions for transferring ownership. This has the advantage that our tokens (i.e. the data points) can be traded by any client implementing this standard, such as wallets or exchanges. By transferring ownership, data points can pass from owner to owner, leaving a trail of provenance records created by each owner along the way. This is useful for implementing provenance applications which aim at creating a lineage (see Section 3), e.g. in supply chain scenarios and business processes [23, 24]. The generic provenance contract complies to the standard by inheriting from an existing ERC721 implementation provided by OpenZeppelin.[3]

4.3.2 Associating provenance records with data points

The generic provenance layer further links data points and their respective provenance records. The framework provides information about associated provenance records of specific data points. Within the generic provenance layer, this is achieved by using a mapping from a data point ID to a set of associated provenance IDs:

$$addr\ (dp) \mapsto \{addr\ (prov_1\ (dp)), addr\ (prov_2\ (dp)), ...\}$$

All associated provenance records ($prov_1$, $prov_2$, etc.) represent parallel provenance traces for the same data point. Of those records, each one represents a completely independent trace of provenance data. For instance, one trace of provenance records might store the temperature history of a physical good, while another one stores its location.

Listing 2 displays an excerpt of the smart contract implementing this layer and demonstrates the general workflow for creating new provenance records. First, the contract verifies that a given token (i.e. data point) exists (Line 7) and that the token belongs to the sender of the message (Line 8). After validating the input provenance records (Line 9), the contract creates a new provenance

```
1  contract GenericProvenance is ERC721, Storage {
2  ...
3  mapping (uint => uint[]) internal associatedProv;
4  ...
5  function createProvenance(uint tokenId, ...)
6  internal returns (uint index) {
7      require(exists(tokenId));
8      require(ownerOf(tokenId) == msg.sender);
9      checkValidProvenance(inputProvenance);
10     uint provId = getProvId();
11     addAssociatedProvenance(tokenId, provId);
12     return super.createProvenance(provId, ...);
13 }
14 ...
```

Listing 2 Generic provenance contract (excerpt)

ID (Line 10), adds it to the list of associated provenance (Line 11), and calls the storage contract's *createProvenance* function (Line 12). Note that the *createProvenance* function of the generic contract is again internal, and thus not accessible publicly. This enables specific provenance contracts which implement concrete use cases to further adapt the functionality according to their needs.

4.4 Specific provenance layer

Smart contracts within this layer utilise the functionality provided by the generic provenance layer, but control a subset of parameters by themselves. This way, use cases can customise the provenance model according to their needs, and control access to the functionality provided by the storage and generic provenance layers. Listing 3 displays an excerpt of an exemplary smart contract implementing this layer.

The provenance model can be customised by defining the *context* parameter, so that it presents the provenance data needed for a specific use case scenario. For instance, in the case of vaccine supply chains, the context should contain temperature and location information about individual vaccines. Hence, a specific contract could define its own *createProvenance* function that requires parameters like *temperature*, and *location* (Lines 7ff) which are

Listing 3 Specific provenance contract (example)

```
1  contract Specific is GenericProvenance {
2  function requestToken()
3  public returns (uint tokenId) {
4      uint tokenId = getTokenId();
5      return super.mint(msg.sender, tokenId);
6  }
7  function createProvenance(uint tokenId,
8  string location, uint temperature, ...)
9  public returns (uint index) {
10     string context = createContext(location, temperature, ...);
11     return super.createProvenance(provId, context, ...);
12 }
13 ...
```

Listing 3 Specific provenance contract (example)

[2] https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md

[3] https://openzeppelin.org/

then combined to form the *context* to be passed on to the *createProvenance* function of the generic provenance contract (Lines 10ff).

Access control happens on two levels. First, a specific contract defines which parts of the generic provenance layer's API are exposed. For instance, even though the generic provenance layer could permit updating or deleting (i.e. invalidating) provenance records, this could be unwanted behaviour in the specific use case at hand. In this case, the contract in the specific provenance layer simply "hides" the functionality, i.e. does not expose it publicly.

Second, access control is relevant for controlling the ownership of data points. Since data point ownership is the decisive factor with regard to who can create provenance records for which particular data points, contracts in the specific provenance layer are responsible for actually assigning ownership, i.e. which tokens get assigned to which clients. For simplicity, the example shown in Listing 3 automatically assigns new tokens to requesting clients (Lines 2ff). However, ultimately, the most suitable approach is dictated by the specific use case at hand. Some further possibilities for assigning ownership are listed below:

- One possibility is for clients to purchase tokens. This has the advantage that the framework is publicly available without the risk of spamming attacks since the acquisition of tokens incurs financial cost. Essentially, the token acquisition cost needs to be low enough for honest users to be willing and able to participate, but high enough to discourage spammers. Of course, such an approach cannot prevent malicious actors with sufficient purchasing power.
- In another approach, the layer manages a white list of authorised clients allowed to request new tokens. This way, the list controls exactly who participates in the provenance system. However, the question arises who is responsible for managing the white list of authorised clients. As discussed above, the proposed framework could also be implemented for private blockchains. In this case, the white list approach is surely the most promising one, since simply all participants in the private blockchain could be white-listed.
- Yet another approach is a list of pending requests. A client sends a transaction requesting tokens. An administrator gets notified about the new request and decides whether to accept or deny the request by assigning or not assigning corresponding tokens. While this approach allows a more fine-grained control over whose requests get accepted and whose requests get denied, it bears the risk of a high degree of centralisation due to the administrator having full control over distributing tokens.

5 Evaluation

We evaluate the framework with regard to the functional and non-functional requirements defined in Section 3. The use cases defined in Section 2 act as basis for the evaluation of Reqs. (1)–(3). In addition, we reason about the fulfilment of Reqs. (4)–(7) in an exemplary fashion applying the use case of vaccine supply chains. However, the scenario of vaccine supply chains is merely used to provide a more descriptive analysis. The information specific to the vaccine supply chain can be substituted with data reflecting any other use case. While the presented framework is blockchain-agnostic, the non-functional requirements (Reqs. (8)–(11)) are evaluated using the proof-of-concept implementation. Experiments were performed on the public Ethereum test networks Rinkeby[4] and Ropsten[5] between 27 November and 13 December 2018. Rinkeby and Ropsten are chosen as test networks since their average block times most closely resemble the block times of the main Ethereum network [25]. Rinkeby and Ropsten use the Proof of Authority (PoA) and Proof of Work (PoW) consensus mechanisms, respectively.

1. *Provenance abstraction* The presented framework consists of multiple abstraction layers. The storage layer is responsible for low-level storage of provenance records while the generic provenance layer extends the storage layer's functionality with generic provenance features and enhanced access control. The generic provenance layer can be extended by use case-specific smart contracts controlling certain parameters of the application, e.g. by assigning ownership of tokens, and/or by exposing or hiding parts of the generic layer's API. For instance, in the scenario of vaccine supply chains, a specific provenance smart contract might give an entity such as the World Health Organisation complete control over who is able to acquire tokens, e.g. only trusted vaccine manufacturers. In the use case of health monitoring systems, multiple approved manufacturers might have control over assigning ownership of data point IDs. In the use case of digital forensics, each newly purchased vehicle might automatically get assigned a new token representing that vehicle. Besides creating provenance records itself, the vehicle could give temporary permission to garages or other entities to create provenance records on its behalf, e.g. to register the date and specifics of an inspection. Hence, we conclude that the framework acts as a base abstraction which can be extended to fulfil the needs of

[4] https://rinkeby.etherscan.io/

[5] https://ropsten.etherscan.io/

specific provenance use cases. Thus, we regard Req. (1) as fulfilled.

2. *High-level and low-level provenance* The framework identifies each data point by its corresponding token. As long as a unique token (i.e. ID) gets assigned to a data point, provenance data for that data point can be recorded. Hence, the framework does not pose any restrictions on the nature of the data point. It is possible to record provenance data for high-level as well as low-level data points. The *context* parameter of a provenance record can be used to add any kind of information the user desires. In the use case of vaccine supply chains, provenance data could not only be collected for the vaccines themselves, but also for sensor readings along the supply chain, e.g. temperature readings which document an uninterrupted cold chain. Within health monitoring systems, the patients themselves might be represented by data points. The patients' provenance traces are then augmented by provenance data from low-level data points deriving from medical devices. Similarly, in the use case of autonomous vehicles, the vehicles themselves are represented by data points and their provenance records get augmented by data points deriving from low-level devices, such as a speedometer or outside temperature sensors. Hence, we regard Req. (2) as fulfilled.
3. *Completeness* The definition of completeness is largely dependent on the specific provenance use case at hand. In the example of vaccine supply chains, all information needed to prove that the cold chain has not been interrupted and that vaccines come from a trusted source need to be recorded. Regarding health monitoring systems, we need to be able to record all information necessary to provide reliable root cause information for medical emergencies. With regard to autonomous vehicles, the system needs to be able to record all information necessary to resolve any disputes between parties involved in an accident. The flexibility of the *context* parameter defined by the provenance model allows the collection of all provenance data relevant for each data point. Therefore, as each individual provenance record can be regarded as complete, the framework provides the necessary foundation for complete provenance tracking (Req. (3)). However, to truly reconstruct the creation and modification history of a data point, not only must the individual provenance records of the data point be complete, but also the complete trace of provenance records regarding the data point should be available. Therefore, in error-prone environments like the IoT where frequent network or device errors may occur, individual use cases need to make sure that sufficient amounts of provenance records reach the provenance framework in order to enable truly complete provenance tracking.
4. *Creation of lineage* As an example, we assume that the manufacturer *SaferVaccines Inc.* produces a new vaccine $vacc_1$. The freshly produced vaccine is packaged with an RFID tag and assigned with a unique ID. A completely new provenance record for the vaccine is created, such as $prov(vacc_1) = \langle addr(vacc_1), \emptyset, \langle agent = operator_1@SaferVaccinesInc, time = 5am, ...\rangle\rangle$. Vaccine $vacc_1$ is loaded onto an aircraft air_1. An RFID reader at the factory gate scans the vaccine and registers a new provenance record of the vaccine leaving the factory, such as $prov(vacc1)' = \langle addr(vacc_1), \{prov(vacc_1)\}, \langle agent = rfid_1@SaferVaccinesInc, time = 5am, ...\rangle\rangle$. Shortly after, an RFID reader at the aircraft's entrance registers the vaccine entering the aircraft, e.g. $prov(vacc_1)'' = \langle addr(vacc_1), \{prov(vacc_1)'\}, \langle agent = rfid_1@air_1, time = 5am, ...\rangle\rangle$. The provenance records $prov(vacc_1)$, $prov(vacc_1)'$, $andprov(vacc_1)''$ represent the lineage of the vaccine. This shows exemplarily how the framework fulfils Req. (4).

(5) *Derivation* Continuing the example from Req. (4), inside the aircraft, sensors constantly monitor the temperature. There are three readings from a temperature sensor: $dp_1 = 38°F$, $dp_2 = 45°F$, $dp_3 = 40°F$. The provenance records are given as follows:

$$prov(dp_1) = \langle addr(dp_1), \emptyset, \langle agent = sensor, time = 7am, ...\rangle\rangle$$
$$prov(dp_2) = \langle addr(dp_2), \emptyset, \langle agent = sensor, time = 8am, ...\rangle\rangle$$
$$prov(dp_3) = \langle addr(dp_3), \emptyset, \langle agent = sensor, time = 9am, ...\rangle\rangle$$

A fourth data point dp_4 is created by a software agent calculating the average temperature: i.e. $dp_4 = (dp_1 + dp_2 + dp_3)/3 = 41°F$. This data point's provenance record is defined as $prov(dp_4) = \langle addr(dp_4), \{prov(dp_1), prov(dp_2), prov(dp_3)\}, \langle agent = averager@air_1 time = 10am, ...\rangle\rangle$. Hence, the framework allows for the creation of provenance records for data points deriving from multiple other data points (Req. (5)).

6. *Provenance for modifications of data points* In a further step, the calculated average temperature dp_4 is converted into a different unit: $dp'_4 = 5\ °C$ (i.e. from Fahrenheit to Celsius). The resulting provenance record looks like $prov(dp'_4) = \langle addr(dp'_4), \{prov(dp_4)\}, \langle agent = converter@air_1, time = 11am, ...\rangle\rangle$. Thus, the

framework is also able to support the modification of data points (Req. (6)).

7. *Parallel provenance* Besides measuring the temperature inside the aircraft (air_1), we might also want to track the location of the aircraft at the same time. The framework enables the creation of parallel provenance records since each data point is mapped to a list of associated provenance records (see Section 4.3). We can create one provenance record $prov_{temperature}(air_1)$ representing the latest temperature inside the aircraft, and one provenance record $prov_{location}(air_1)$ representing the latest location of the aircraft. Hence, we regard (Req. (7)) as fulfilled.
8. *Integrity* The presented framework uses blockchain technology to provide trustless and tamper-proof storage of provenance records for IoT data. Thus, once records have entered the system, the integrity of records depends on the underlying blockchain technology. Ethereum is a public blockchain with around 7075 fully validating nodes securing the network at the time of writing.[6] Because of its PoW consensus mechanism and the high number of fully validating nodes, we consider the Ethereum network as secure. Hence, we consider the integrity requirement for data and computations on Ethereum as fulfilled. However, the framework also needs to provide mechanisms to ensure that only correct records enter the system in the first place. As a first step, we implement the concept of ownership of data points using tokens to prevent arbitrary clients from creating provenance records. In future work, this concept could be further extended by mechanisms that guarantee the correctness of the records themselves, e.g. via physical unclonable functions (PUFs) [2].
9. *Availability and fault tolerance* In public blockchain networks like Ethereum, potentially any client can run a full node giving the client access to the complete state of the blockchain. Furthermore, blockchains are designed from the ground up to provide Byzantine fault tolerance. That is, blockchains continue to operate even if some nodes fail or act maliciously with imperfect information about these failures or bad actors. Hence, we consider (Req. (9)) as fulfilled.
10. *Privacy* While the privacy requirements of a provenance system largely depend on the concrete use case at hand, in scenarios with sensitive data, such as a health monitoring system, privacy is crucial [13]. However, privacy remains an ongoing challenge in the realm of (public) blockchains, since a blockchain's security relies on data being transparent and verifiable by every participant. A blockchain-based data provenance framework suffers from the same problem. While some propose to use private blockchains in scenarios where privacy is important [13], this is not sufficient in scenarios which also depend on the system being publicly available, such as global vaccine supply chains. Possible solutions to overcome the privacy limitations of blockchain-based applications in the context of the IoT are discussed in Section 6.
11. Scalability

    Regarding scalability, we evaluate two aspects. First, we examine the gas consumption of the system. Second, we measure transaction throughput and latency.

Gas consumption Creating a new provenance record should have the same gas consumption whether there are thousand already existing provenance records in the system or there are none. Since our framework stores provenance records as a mapping $addr(prov) \mapsto prov$, we can access records directly via their ID. Hence, creating, updating, or accessing provenance records is a constant gas operation. Furthermore, we analyse how the size of the input parameters influences the gas consumption for creating provenance records. We examine the creation rather than the updating or deletion of provenance records since these operations are generally less expensive than the creation operation. Evaluating the gas consumption for creation thus gives a good estimate on the upper limit of gas consumption of the framework. Figures 5 and 6 show the gas consumption with regard to the size of the context parameter and the number of input provenance records, respectively. When increasing the size of the context parameter, a stepwise ascent in gas consumption is visible in Fig. 5. As expected, passing a longer context string when creating a provenance record causes greater gas consumption, since more storage

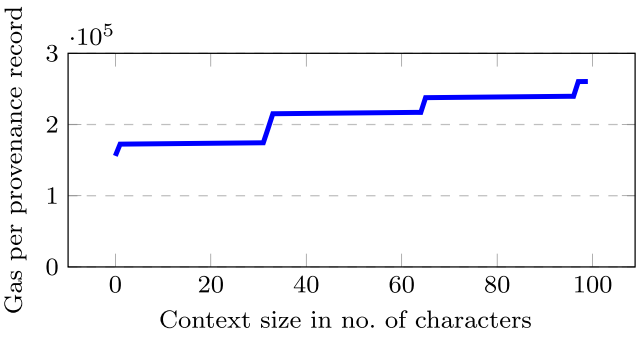

Fig. 5 Gas consumption for provenance creation with respect to an increasing context parameter

[6] 22 October 2019 (https://ethernodes.org/)

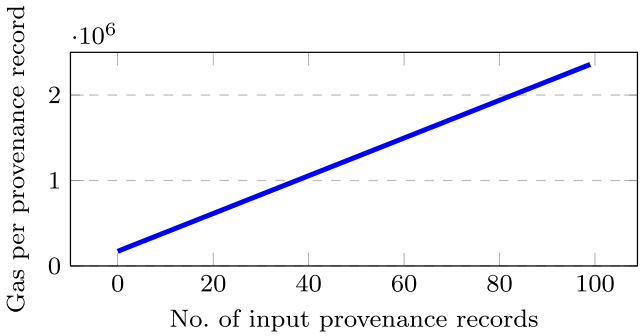

Fig. 6 Gas consumption for provenance creation with respect to an increasing number of input provenance records

is needed. Similarly, when increasing the number of input provenance records, gas consumption rises linearly as can be seen in Fig. 6, though more rapidly in comparison to increasing the context size. This rapid ascent in gas consumption stems from the fact that users are only allowed to add input provenance records that are actually existent and valid. Thus, when creating new records, the framework checks, for every input provenance record, if that record exists and if it is valid. This check can only happen in a for-loop which is an expensive operation in Ethereum. Because of the increases in gas consumption caused by these two input parameters, it makes sense for concrete provenance use cases to implement limits regarding both parameters.

Latency The latency of creating provenance records is evaluated by averaging 100 measurements of the time passing between the moment a transaction is submitted to the network and its execution and inclusion in a block. In general, two factors influence transaction latency, namely the overall network load and the gas price. With a higher number of pending transactions (i.e. a higher network load), not every transaction can be included within the next immediate block which means higher priced transactions get prioritised. We repeat the experiment with gas prices of 1, 2, 5, 10, 20 and 50 GWei (1 GWei = 0.000000001 ETH) to see how the gas price influences transaction latency. While performing the experiments, we observe the test networks to make sure that overall network load remains relatively stable.

Figure 7 shows the average latency and standard deviation for creating provenance records. Latency on the Ropsten test network is the highest (average transaction latency between 15–36 s), followed by Rinkeby (14–35 s). The standard deviation for Ropsten is also quite high with values of about 11–29. On average, transactions with gas prices of only 1 Gwei take longer to be confirmed than transactions with gas prices of 2 Gwei and beyond. As the gas price is increased, the average latency converges to the average block times of the respective networks, since higher priced transactions are almost certainly included within the next immediate block. This also explains the higher standard deviations of Ropsten in comparison with Rinkeby. Rinkeby uses PoA, that means a new block gets consistently mined every 14–15 s, resulting in lower standard deviations. In Ropsten, on the other hand, blocks get mined if a node can solve the cryptographic puzzle of the PoW. Since this happens at different intervals, Ropsten exhibits higher standard deviations.

Transaction throughput To test throughput, we attempt to submit up to 150 transactions to the test network during a 60-s time frame. Afterwards, we divide the number of confirmed transactions with the time frame of 60 s to calculate the transactions per second (TPS). This is repeated 20 times to calculate the mean and standard deviation. The experiment is repeated with different gas prices (1, 2, 5, 10, 20 Gwei) on the two different test networks.

The results of the experiment are displayed in Fig. 8. Generally, throughput for the Rinkeby test network is the greatest with about 1.6 TPS for gas prices of 1 Gwei and beyond. On Ropsten, we see a throughput of about 1–1.2 TPS for gas prices of 2 Gwei and beyond. We explain the smaller average throughput of the Ropsten test network by the total load of each network during the experiments which was generally higher on Ropsten than on Rinkeby. Note that throughput was measured from a user perspective, i.e. transactions were submitted from a single account. The

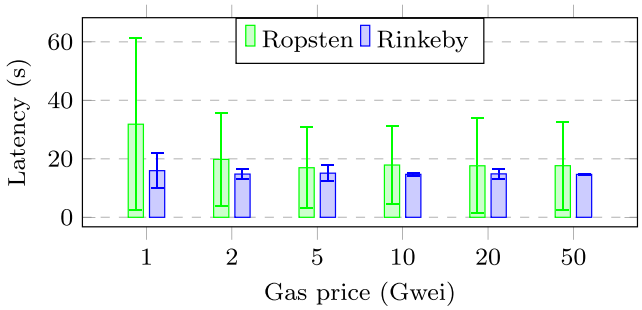

Fig. 7 Avg. latency for creating provenance records

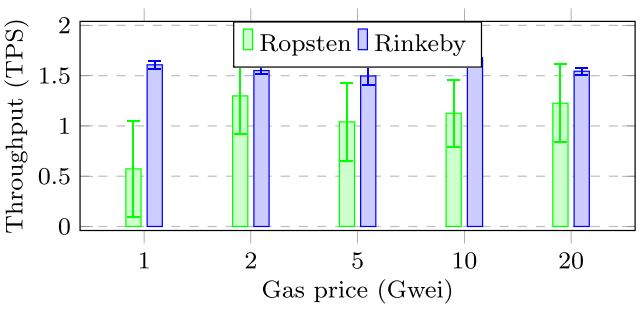

Fig. 8 Avg. throughput for creating provenance records

maximum global throughput of the Ethereum blockchain is roughly 20 TPS.

To sum up, the framework is able to fulfil all functional requirements defined in Section 3. Additionally, by leveraging blockchain technology, the non-functional requirements of integrity and availability can be fulfilled. Regarding scalability and privacy, the use of blockchain technology poses limitations. While the system scales with regard to the number of already existent provenance records, the throughput and latency of the framework are constrained. Depending on the number of provenance records that need to be recorded in a given time frame, the limited throughput and latency can pose problems. For instance, while in the use case of vaccine supply chains sensor readings might only occur every so often, the use case of autonomous vehicles might require multiple sensor readings per second, leading to high throughput requirements for the provenance system. Furthermore, the inherent transparency of blockchain technology naturally poses a challenge for privacy-sensitive applications such as in health monitoring solutions.

6 Discussion

As we have demonstrated in Section 5, the presented framework is able to satisfy Reqs. (1)–(7) by building on top of the generic data provenance model for the IoT and by providing abstraction layers which specific use cases can adopt and extend to fit their own requirements such as the assignment of ownership of data point IDs. Furthermore, the framework relies on blockchain technology to provide guarantees of integrity and availability (Req. (8) and Req. (9)). However, the fundamental design of what makes the blockchain secure causes its limited scalability and privacy [6]. While in some cases, permissioned blockchains like Hyperledger Fabric[7] can be employed to provide better privacy and scalability [13], in global IoT scenarios entailing massive amounts of data and frequent data updates where public access to the system is necessary, permissioned blockchains might not be a viable option [26]. Hence, a solution is needed to improve the privacy and scalability properties of permissionless blockchain systems while keeping its integrity and availability guarantees. Several recent blockchain developments try to tackle these limitations, e.g. state channels [27], zero-knowledge proofs [28], and sidechains [29].

[7] https://www.hyperledger.org/projects/fabric

6.1 State channels

State channels increase scalability by off-chaining state transitions [27]. Two participants open a state channel by locking some state on the blockchain (e.g. in a smart contract). Once locked, they perform transactions on that state off-chain. The only requirement is that both participants sign off these transactions, using signatures verifiable on-chain, and that the transactions are ordered. This allows both participants to finalise the state on the blockchain at any time since they can prove via the signatures that the counterpart agreed to the update. The guarantee of settling a transaction at any point is as good as having the transaction executed directly on the blockchain. This way, state channels reduce the cost of transactions since only the finalisation step occurs on-chain. As only the final state update becomes visible in the blockchain, state channels can further be used to hide any intermediate transactions from outsiders [27]. These characteristics make state channels a viable choice for increasing scalability and improving privacy in blockchain-based IoT applications, e.g. in scenarios with micro-transactions occurring between devices. Note that implementations of state channels range from application-specific (i.e. payment channels [30]) to generic [31] and from unidirectional to bidirectional. In this discussion, we consider generic, bidirectional state channels.

6.2 Zero-knowledge proofs

Zero-knowledge proofs are an off-chaining pattern, where a prover tries to convince a verifier about the correct execution of a computation without revealing the parameters of the computation to the verifier [27]. The aim is for the verifier to be able to verify that the prover correctly executed the computation without the verifier knowing any of the inputs required for the computation, thus having zero knowledge. To take advantage of zero-knowledge proofs in blockchains, zkSNARKs (zero-knowledge Succinct Non-interactive ARguments of Knowledge) can be employed [28]. This serves two purposes: resource-intensive computations can be performed and verified off-chain and the input of computations can be kept private.

However, proof construction in zkSNARKs is computationally expensive, leading to high hardware requirements for the prover. This can be a problem in IoT scenarios if the prover is hardware-constrained. Also, off-chaining state transitions is only beneficial when the verification of the computation is less expensive than the computation itself. Thus, while zkSNARKs can improve scalability and privacy, it needs to be evaluated whether the initial overhead is justifiable in IoT applications.

6.3 Sidechains

The main idea of sidechains is to take load off of an existing blockchain (i.e. the parent chain) by outsourcing some of its data and computation to a separate blockchain (i.e. the sidechain) [29]. Sidechains are "pegged" to the parent chain, i.e. the state of the sidechain is secured via regular commits to the parent chain. Such commits entail the root hash of a Merkle tree consisting of all transactions that have occurred on the sidechain. This way, all transactions of the sidechain are secured via a single hash on the parent chain. This link enables participants to retreat back to the secure and reliable parent chain via so-called exits in case malicious behaviour is detected on the sidechain [29], e.g. when sidechain validators mine invalid blocks or withhold blocks completely. While sidechains can potentially employ different, more efficient consensus mechanisms than the parent chain, scalability gains primarily derive from participants of a sidechain not having to validate all global transactions of the parent chain anymore, but only the transactions occurring on the sidechain. Thus, a validator's hardware requirements can remain low, while transaction throughput is increased as transactions and data of the parent chain are split into multiple different sidechains. As such, sidechains improve throughput rather than latency [32]. Furthermore, sidechains could be kept secret to improve privacy.

The three approaches discussed above are promising with regard to improving scalability and privacy of blockchain applications. In future work, it needs to be evaluated if and how these solutions can be employed to provide scalability and privacy improvements in global IoT scenarios.

7 Related work

As mentioned in Section 1, there are existing studies which focus on providing a generic blockchain-based data provenance solutions for the IoT [8, 9]. The framework presented by Baracaldo et al. addresses the following requirements [9]: tamper prevention (i.e. ensuring integrity of provenance data), high availability, fine-grained access control, and enabling resource-constrained devices to participate in the framework. However, an actual implementation is not provided. Similarly, while Polyzos et al. mention key security requirements of their blockchain-assisted information distribution system for the IoT [8], the approach is described only on a conceptual level and no implementation is given. In contrast, the framework presented in the work at hand has been fully implemented and is available as open-source software.

Further solutions related to blockchain-based data provenance in the realm of the IoT go beyond conceptual approaches [10–14]. The approaches presented in [10–12] focus on securing provenance data in supply chains. The framework proposed in [11] uses a tokenised recipe model to track the relationship between resources and products, the authors of [10] propose a blockchain-based system to track ownership of products in the post supply chain, and Wu et al. [12] discuss an architecture of permissioned and public ledgers to comply with the privacy requirements of trading partners. Griggs et al. present a blockchain-based patient monitoring system to provide notifications in case of medical emergencies [13]. The approach utilises smart health devices and a private blockchain to provide secure analysis of sensor data. Cebe et al. present a digital forensics framework based on blockchain technology that tracks information about vehicles, which, in case of accidents, can be used for resolving disputes between drivers, insurance companies, and maintenance service providers [14]. In general, the solutions presented in [10–14] all demonstrate how provenance concepts can be brought onto the blockchain (e.g., via smart contracts). However, in contrast to our work, though IoT technologies are addressed, these solutions are use case–specific and do not provide a generic provenance framework as is required because of the heterogeneous nature of the IoT [1, 7].

Provenance in the supply chain has also received attention in the industry. Startups like Provenance[8] and Everledger[9] focus on securely tracking goods traveling along a supply chain via blockchain technology. However, in contrast to our work, their sole focus lies on tracking physical items instead of any kind of data within the IoT.

Furthermore, works have also focused on blockchain-based data provenance solutions outside the IoT domain [33–36]. Liang et al. present Provchain [33], a system that provides the ability to record and verify the operation history of data on the cloud. Whenever an operation (read/write) happens on a file in the cloud, the cloud service provider stores a provenance record locally and anchors the hash of the record to a public blockchain. This way, a timestamp proof of that particular operation exists which is verifiable in the future. The information saved on the blockchain is used for the verification of a provenance record; however, it does not provide any provenance data itself. While users can verify that certain operations on a file have occurred, they still depend on a central entity to provide them with a complete set of provenance data for a particular file. Tosh et. al propose BlockCloud [34], a data provenance solution in the cloud based on a Proof of Stake (PoS) blockchain to overcome scalability limitations of blockchains based on PoW. While the architecture of

[8] https://www.provenance.org/

[9] https://www.everledger.io/

BlockCloud is outlined, in contrast to our work, no concrete implementation details are provided.

Secure provenance for scientific data is in the focus of [35]. Here, the authors present DataProv, a system which securely captures scientific provenance data. DataProv uses the OPM data model [20] for recording provenance data. The system provides an access control mechanism built on smart contracts on the Ethereum blockchain platform to control changes of documents in the cloud while taking the actual verification process of document changes off the chain. This privacy-focused solution looks promising also for IoT-related domains. However, while using the OPM data model is fine for scientific contexts, in IoT contexts, the utilization of a simpler, more fine-grained provenance model could be useful [7].

A completely different solution is offered by Ruan et al. [36]. Here, the authors provide LineageChain, a solution that captures fine-grained provenance for blockchains. Smart contracts that depend on some old state of the blockchain for execution can query LineageChain for historical state information in a tamper-evident way. However, LineageChain requires a modification of the underlying blockchain protocol itself and thus cannot be leveraged on existing blockchain platforms.

As can be seen by the discussion, to the best of our knowledge, there is no other approach which provides a generic, blockchain-based framework for tracking data provenance in the IoT.

8 Conclusion

In this paper, we have presented a blockchain-based framework for providing generic data provenance functionality in the IoT. The framework enables the adoption and implementation of data provenance concepts for various IoT use cases. In a first step, we have identified several functional and non-functional requirements which need to be fulfilled for creating a secure data provenance solution appropriate for the heterogeneous nature of the IoT. Afterwards, we have demonstrated how a generic IoT data provenance model together with a layered architecture of smart contracts can be utilised to satisfy the functional requirements. Furthermore, by leveraging blockchain technology, the framework fulfils the non-functional requirements of integrity and availability but has some limitations regarding scalability and privacy. As we have seen in the discussion, multiple current blockchain developments seek to overcome these issues without jeopardising blockchain security, namely state channels, zero-knowledge proofs, and sidechains. Thus, in our future work, we will evaluate in detail to what extent these approaches provide solutions to the privacy and scalability issues in the context of blockchain-enhanced IoT applications, such as our data provenance framework.

Funding Information Open access funding provided by TU Wien (TUW).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Li S., Da Xu L., Zhao S. (2015) The internet of things: a survey. In: Information systems frontiers, vol 17.2, pp 243–259
2. Aman M. N., Chua K. C., Sikdar B. (2017) Secure Data Provenance for the Internet of Things. In: 3rd ACM international workshop on IoT privacy, trust, and security. ACM, pp 11–14
3. Bertino E. (2014) Data trustworthiness–approaches and research challenges. In: Data privacy management, autonomous spontaneous security, and security assurance. Springer, pp 17–25
4. Freire J., et al. (2008) Provenance for computational tasks: a survey. In: Computing in science & engineering, vol 10.3, pp 11–21
5. Hasan R., Sion R., Winslett M. (2009) Preventing history forgery with secure provenance. In: ACM transactions on storage, vol 5.4. Article no. 12
6. Christidis K., Devetsikiotis M. (2016) Blockchains and smart contracts for the internet of things. In: IEEE Access, vol 4, pp 2292–2303
7. Olufowobi H, et al. (2017) Data provenance model for Internet of Things (IoT) systems. In: Service-oriented computing-ICSOC 2016 workshops: revised selected papers. Springer, Berlin, pp 85–91
8. Polyzos G. C., Fotiou N. (2017) Blockchain-assisted information distribution for the Internet of Things. In: 2017 IEEE international conference on information reuse and integration. IEEE, pp 75–78
9. Baracaldo N, et al. (2017) Securing data provenance in Internet of Things (IoT) systems. In: Service-oriented computing-ICSOC 2016 workshops: revised selected papers. Springer, Berlin, pp 92–98
10. Toyoda K., et al. (2017) A novel blockchain-based product ownership management system (POMS) for anti-counterfeits in the post supply chain, vol 5, pp 17465–17477
11. Westerkamp M., Victor F., Küpper A. (2018) Blockchain-based supply chain traceability: token recipes model manufacturing processes. IEEE, pp 1595–1602

12. Wu H., et al. (2017) A distributed ledger for supply chain physical distribution visibility. In: Information, vol 8.4, pp 137:1–137:18
13. Griggs K. N., et al. (2018) Healthcare blockchain system using smart contracts for secure automated remote patient monitoring. In: Journal of medical systems, vol 42.7, pp 130:1–130:7
14. Cebe M., et al. (2018) Block4forensic: an integrated lightweight blockchain framework for forensics applications of connected vehicles. In: IEEE communications magazine, vol 56.10, pp 50–57
15. Sigwart M., et al. (2019) Blockchain-based data provenance for the Internet of Things. In: 9th international conference on the internet of Things (IoT 2019). ACM, pp 15:1–15:8
16. Simmhan YL, Plale B, Gannon D (2005) A survey of data provenance tech- niques. Tech. rep. IUB-CS-TR618. Computer Science Department, Indiana University, Bloomington, Indiana, USA
17. Comes T, Bergtora Sandvik K, Van de Walle B (2018) Cold chains, interrupted: the use of technology and information for decisions that keep humanitarian vaccines cool, vol 8.1, pp 49–69
18. Megget K Massive fake vaccine racket busted in Indonesia https://www.securingindustry.com/pharmaceuticals/massive-fake-vaccine-racket-bustedin-indonesia/s40/a2849/#.WrylEtNuaL4. Accessed 30 March 2018
19. Cooper K (2020) Hyperledger Burrow. https://wiki.hyperledger.org/display/burrow/Burrow+-+The+Boring+Blockchain
20. Moreau L., et al. (2011) The open provenance model core specification (v1.1). In: Future generation computer systems, vol 27.6, pp 743–756
21. Moreau L., et al. (2013) PROV-DM: the PROV data model. https://www.w3.org/TR/prov-dm/. Accessed 9 Jul 2018
22. Cryptographic Tokens. https://blockchainhub.net/tokens/. Accessed 29 May 2019. (2019)
23. Prybila C., et al. (2020) Runtime verification for business processes utilizing the bitcoin blockchain. In: Future generation computer systems, vol 107, pp 816–831
24. Mendling J., Weber I., Aalst W. V. D., Brocke J. V., Cabanillas C., Daniel F., Debois S., Ciccio C. D., Dumas M., Dustdar S., et al. (2018) Blockchains for business process management–challenges and opportunities. In: ACM Transactions on Management Information Systems (TMIS), 9(1), pp 1–16. ACM New York, NY, USA
25. Ethereum StackExchange (2018) Comparison of the different TestNets. https://ethereum.stackexchange.com/questions/27048/comparison-of-the-different-testnets
26. Fernández-Caramés T. M., Fraga-Lamas P. (2018) A review on the use of blockchain for the Internet of Things. In: IEEE Access, vol 6, pp 32979–33001
27. Eberhardt J., Tai S. (2017) On or off the Blockchain? Insights on off-chaining computation and data. In: 2017 European conference on service-oriented and cloud computing. Springer, pp 3–15
28. Eberhardt J, Tai S (2018) ZoKrates – scalable privacy-preserving off-chain computations. In: IEEE international conference on blockchain. IEEE, pp 1084–1091
29. Back A et al (2014) Enabling blockchain innovations with pegged sidechains. https://blockstream.com/sidechains.pdf. Accessed 10 April 2019
30. Lightning Network. https://lightning.network/. Accessed 30 Oct 2019
31. Dziembowski S, Faust S, Hostáková K (2018) General state channel networks. In: 2018 ACM SIGSAC conference on computer and communications security. ACM, pp 949–966
32. Konstantopoulos G (2018) Plasma cash: towards more efficient plasma constructions. https://github.com/loomnetwork/plasma-paper/blob/master/plasmacash.pdf. Accessed 6 Sept 2018
33. Liang X, et al. (2017) Provchain: a blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In: 17th IEEE/ACM international symposium on cluster, cloud and grid computing. IEEE, pp 468–477
34. Tosh D, et al. (2019) Data provenance in the cloud: a blockchain-based approach. In: IEEE consumer electronics magazine, vol 8.4, pp 38–44
35. Ramachandran A, Kantarcioglu M (2018) SmartProvenance: a distributed, blockchain based dataprovenance system. In: 8th ACM conference on data and application security and privacy. ACM, pp 35–42
36. Ruan P, et al. (2019) Fine-grained, secure and efficient data provenance on blockchain systems. In: Proceedings of the VLDB endowment, vol 12.9, pp 975–988