Future Generation Computer Systems 157 (2024) 164–179

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Enabling privacy-aware interoperable and quality IoT data sharing with context

Tek Raj Chhetri [a,b,c,d,*], Chinmaya Kumar Dehury [e], Blesson Varghese [f], Anna Fensel [g,h], Satish Narayana Srirama [i], Rance J. DeLong [j]

[a] McGovern Institute for Brain Research, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA
[b] Semantic Technology Institute (STI), Department of Computer Science, Universität Innsbruck, Innsbruck, 6020, Austria
[c] Center for Artificial Intelligence (AI) Research Nepal, Sundarharaincha-09, Nepal
[d] Web and Internet Science Research Group, School of Electronics and Computer Science, The University of Southampton, University Road, Highfield Campus Southampton SO17 1BJ, United Kingdom
[e] Institute of Computer Science, University of Tartu, Tartu, 50090, Estonia
[f] School of Computer Science, University of St Andrews, St Andrews, Fife, UK KY16 9SX, United Kingdom
[g] Wageningen Data Competence Center, Wageningen University & Research, Wageningen, 6708 PB, The Netherlands
[h] Artificial Intelligence Chair Group, Wageningen University & Research, Wageningen, 6708 PB, The Netherlands
[i] School of Computer and Information Sciences, University of Hyderabad, Hyderabad, 500046, India
[j] The Open Group, Reading, Berkshire, RG1 1AX, United Kingdom

ARTICLE INFO

Keywords:
Data sharing
Edge intelligence
Interoperability
Internet of Things (IoT)
Knowledge graphs
General Data Protection Regulation (GDPR)
Smart cities

ABSTRACT

Sharing Internet of Things (IoT) data across different sectors, such as in smart cities, becomes complex due to heterogeneity. This poses challenges related to a lack of interoperability, data quality issues and lack of context information, and a lack of data veracity (or accuracy). In addition, there are privacy concerns as IoT data may contain personally identifiable information. To address the above challenges, this paper presents a novel semantic technology-based framework that enables data sharing in a GDPR-compliant manner while ensuring that the data shared is interoperable, contains required context information, is of acceptable quality, and is accurate and trustworthy. The proposed framework also accounts for the edge/fog, an upcoming computing paradigm for the IoT to support real-time decisions. We evaluate the performance of the proposed framework with two different edge and fog–edge scenarios using resource-constrained IoT devices, such as the Raspberry Pi. In addition, we also evaluate shared data quality, interoperability and veracity. Our key finding is that the proposed framework can be employed on IoT devices with limited resources due to its low CPU and memory utilization for analytics operations and data transformation and migration operations. The low overhead of the framework supports real-time decision making. In addition, the 100% accuracy of our evaluation of the data quality and veracity based on 180 different observations demonstrates that the proposed framework can guarantee both data quality and veracity.

1. Introduction

The Internet of Things (IoT) ecosystem continues to expand alongside other newer computing paradigms, such as edge/fog computing, leading to exponential growth in data. With the IoT ecosystem, we refer to the integrated layer of IoT hardware, software, and connectivity [1]. The term "big data" is used to refer to these massive amounts of data generated by the IoT [2]. For example, IoT connections are expected to reach 83 billion by 2024, rising from 35 billion in 2020 [3]. As the IoT ecosystem expands, so do the associated complexities and challenges arising from heterogeneity, such as data interoperability, accuracy, and quality. These issues must be addressed as the IoT is implemented in crucial sectors, such as healthcare for remote robot surgery [4].

Fig. 1 provides a high-level overview of the use of IoT in different environments by integrating recent edge/fog computing paradigms and highlights the corresponding challenges. The first challenge is associated with data quality and accuracy. The term "accuracy" is also referred to as "data veracity". Data quality issues occur due to reasons, such as sensor calibration inaccuracies and environmental effects that cause erroneous observations [5–7]. With data quality, we refer to the suitability of data (or fitness to use) for a specific

* Corresponding author at: Semantic Technology Institute (STI), Department of Computer Science, Universität Innsbruck, Innsbruck, 6020, Austria.
E-mail addresses: Tek-Raj.Chhetri@uibk.ac.at, tekraj@mit.edu (T.R. Chhetri), chinmaya.dehury@ut.ee (C.K. Dehury), bv6@st-andrews.ac.uk (B. Varghese), anna.fensel@wur.nl (A. Fensel), satish.srirama@uohyd.ac.in (S.N. Srirama), r.delong@opengroup.org (R.J. DeLong).

https://doi.org/10.1016/j.future.2024.03.039
Received 30 January 2023; Received in revised form 20 March 2024; Accepted 22 March 2024
Available online 27 March 2024

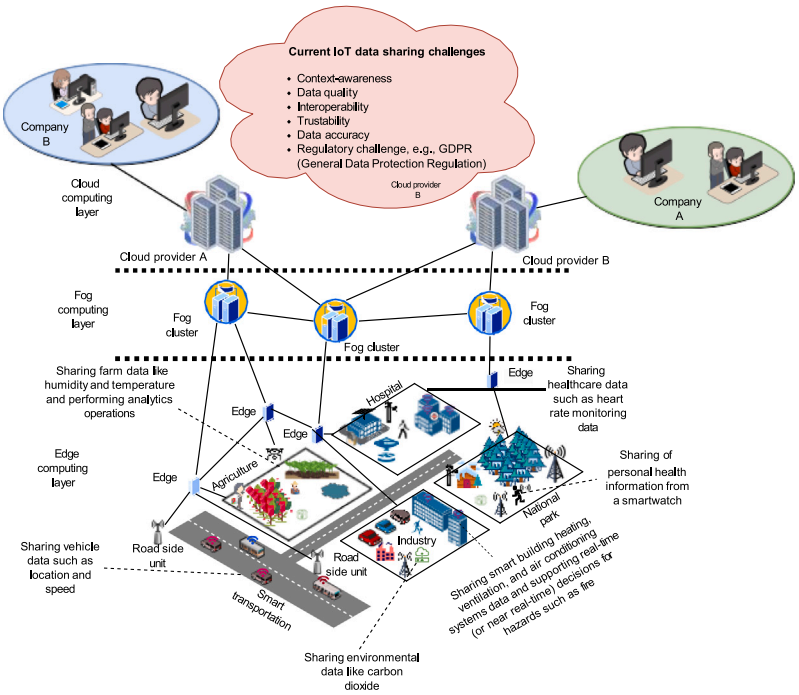

Fig. 1. IoT data sharing environment with an edge/fog scenario and the associated challenges.

application or task [8]. More than 50 studies have been conducted using advanced statistical and machine learning techniques to ensure that the observation values are not erroneous [7]. This is because poor-quality data can lead to poor or flawed decisions. However, such post-hoc approaches as anomaly and outlier detection are expensive and not suitable for real-time. Data integrity is also important and is a key challenge [6,9]. Therefore, there is a need for a scalable solution that intrinsically addresses the data quality challenge and also addresses the data veracity problem. Furthermore, there is a need for understanding data veracity in the context of edge/fog that will enable decision making in smart cities and autonomous driving.

Interoperability and the lack of context are other challenges [2,5,6,9]. Interoperability is the ability of heterogeneous systems to exchange and consume data [10]. Interoperability, for example, accounts for 40% of the potential benefits of the IoT, which is anticipated to have a yearly economic impact of $11.1 trillion by 2025 [11]. Interoperability is essential to enable the exchange and use of information across heterogeneous systems. Context (or contextual information) in the case of data sharing is the set of interrelated conditions describing the data, such as sensor information, units of measurement, and the observed property. This is extremely crucial as the heterogeneity of the IoT grows and its use in automated decision making increases. In addition, the availability of context facilitates accurate decision making. Temperature, for instance, can be measured on a variety of scales; if the unit of measurement, for example, is not specified, the likelihood of errors increases. This is because the same measurement value on different scales represents different meanings. For example, if 53.6 on the Celsius scale is a high temperature, in Fahrenheit it is not, as it would just mean 12 on the Celsius scale. A recent study on data quality and trust in the IoT by Byabazaire et al. [5] found that, despite its importance, context information is often disregarded. According to Byabazaire et al. [5], context is a component of data quality, and no solutions have considered context information while assessing data quality.

Data sharing in the IoT presents additional challenges related to privacy and trust [12,13]. For example, the privacy concern has led to the implementation of legislation to safeguard privacy, such as the General Data Protection Regulation (GDPR) in 2018 [14], which has added an additional layer of complexity. GDPR aims to improve privacy and transparency in data sharing and processing through the notion of informed consent from the data subject (i.e., a natural person) which must be obtained prior to sharing and processing data containing personally identifiable information (PII) concerning the data subject. Non-compliance with GDPR can result in a *maximum fine of up to 20 million euros or 4% of the firm's global annual revenue for the preceding financial year, whichever is greater (Article 83)*. Consequently, privacy is also a concern in the IoT [15] where applications such as smart cities, vehicle data sharing, and healthcare generally involve PII data.

Trust has been extensively studied [5,12]. Over fifty studies have been done on IoT trust management using methods like statistical approaches and also the blockchain, similar to works on data quality [12]. However, they are primarily concerned with the composition and management of IoT services and have not addressed issues such as heterogeneity [12]. Studies on trust focusing on IoT data, such as those by Byabazaire et al. [16], have also been conducted, proposing a framework to derive the trust metric from the shared IoT data. However, Byabazaire et al.'s [16] framework focuses on deriving trust from shared data, as opposed to trust as an intrinsic metric of real-time data sharing. In addition, as pointed out by Byabazaire et al. [16], their work does not address memory or computational costs, which are essential in the case of the IoT due to its resource-constrained nature. Fortineo et al. [13], in their survey on trust and reputation in the IoT, further highlights the need to consider the resource constrained nature of IoT and the newer edge/fog computing paradigms. There is a need for a privacy-aware solution for data sharing that incorporates an intrinsic trust metric.

Therefore, the proposed framework introduces a trust metric intrinsic to the shared. In addition, the proposed framework includes consent checking to ensure privacy and compliance with GDPR. This is because the integration of the trust metric (or score) helps determine the extent to which one should rely on shared data. The same, i.e., the significance and necessity of the trust metric (or score), is emphasized by Byabazaire et al. [5] in their survey on data quality and trust in IoT for secure end-to-end data sharing, along with the difficulties such as how to securely propagate the trust metric. Additionally, trust metrics have been implemented in other contexts, such as Google search [17],

where trustable sources are ranked highly. This further emphasizes the significance of the trust metric.

The proposed framework, which addresses the aforementioned challenges (or problems), such as interoperability, quality, privacy, and trust issues in IoT data sharing in a heterogeneous environment, uses semantic technology. Semantic technology, specifically ontologies and knowledge graphs (KGs), which are based on the linked-data concept, can represent real-world relationships, be used to build knowledge, and can also provide reasoning, interoperability, data enrichment, and data variety handling capabilities [18]. In addition, the proposed approach makes use of the blockchain, which provides properties such as tamperproofness [19] and makes the proposed solution suitable even for an untrustworthy environment.

The proposed framework supports real-time (or near real-time) decision making and utilizes low resources, as evidenced by the following findings: (i) an average CPU utilization at the edge and fog, respectively, of 33.61% and 15.62% for analytics operations and 12.35% and 1.24% for data transformation and migration operations; (ii) an average time to perform analytics operations at the fog layer of 1.1 s and the edge of 7.2 s; and (iii) 51 MB of memory utilization.

The paper is organized as follows. Section 2 provides background information, the motivation for our work, and the contributions. Section 3 provides an overview of the related work. Section 4 provides an overview of methodology used in our study for ontology modeling as well as for weighted trust metric. Section 5 details the proposed system (or framework). Section 6 provides information on the experimental setup and implementation. Section 7 provides details on the experiment and evaluation of our work. Finally, Section 8 provides the conclusion.

2. Background, motivation & contribution

Data sharing and integration are vast and extremely important in today's connected world. However, despite the importance of data sharing, the number of works in this area is limited [20]. Additionally, with the rise of new technology and industrial and legal requirements, the data sharing and integration landscape continuously evolves, introducing new opportunities and obstacles. One such instance is the implementation of the GDPR, which has altered the data sharing paradigm. For example, our previous work [21] that the proposed framework in this paper extends established a data pipeline framework for secure and accurate data migration and does not cover GDPR. Moreover, Yang et al. [20] recently proposed an edge data sharing framework named EdgeShare for industrial IoT data sharing using blockchain. However, as with other works, the other important aspects, such as data quality and legal regulations, still need to be covered to provide a holistic data sharing solution.

2.1. Motivation

The major motivation for this work is the existing challenges, such as data quality and their impact in supporting decision making tasks in domains like healthcare and smart cities. For example, three-quarters of IoT projects fail solely because of the data quality issue [6,7]. In a manner analogous to data quality, the other motivations include challenges, lack of context awareness, trust, accuracy, and interoperability, all of which affect decision making. For example, one can only imagine the repercussions of having data, even a simple temperature measurement, without appropriate context or an interoperability issue in patient monitoring systems. In addition, the limited work in data sharing [20] and the need for a comprehensive solution that addresses the previously-mentioned challenges, a lesson learned from our previous work [22] and the smashHit[1] project, serve as additional motivation.

[1] https://smashhit.eu.

Moreover, it is just as essential to understand the implications of the new technological solutions, particularly with regard to supporting real-time (or near real-time) decision making in edge/fog scenarios, as it is to address the previously mentioned challenges. The reason is the growing use of IoT in industries such as healthcare and manufacturing, which require real-time (or near real-time) decision making. It is the same reason, i.e., the need to support the real-time requirements and to reduce the latency caused by cloud–IoT communication, the edge/fog paradigms were introduced [23]. Understanding the suitability of the proposed solution to support real-time decision making constitutes the another motivation for our work.

In line with the motivations, this research seeks to answer the following research questions:

1. How can we assure accurate and trustworthy data sharing in an interoperable manner in a complex, heterogeneous environment such as a smart city to support data-driven decision making?
2. Can we assure the need for real-time (or near real-time) decision making, i.e., support analytics operation in a constrained environment while addressing other challenges such as interoperability and data veracity?

2.2. Contribution

Based on the motivation of our work, we make the following contributions.

1. The design and implementation of a privacy-aware interoperable IoT data sharing framework based on semantic technology and blockchain that intrinsically guarantees data quality based on predefined criteria and also ensures data veracity.
2. An introduction of the weighted trust metric (see Section 4.2) that is intrinsic to shared data and considered within an edge/fog scenario.
3. Assessing context information as a component of data quality, for which, according to [5], there are currently no solutions.
4. The design and implementation of the framework to provide data analytics capability to support real-time decision making requirements at edge/fog, along with interoperable and quality data sharing.
5. Finally, the performance evaluation of the proposed framework in terms of memory and CPU (Central processing unit) overhead and execution time with two different edge and fog–edge scenarios utilizing resource-constrained IoT devices such as the Raspberry Pi.

3. Related work

In this section, we provide an overview of related works focusing on IoT data sharing. The summary of related works includes studies that address issues such as interoperability, data quality, and accuracy. Additionally, our review includes studies that examine GDPR-compliant IoT data sharing and excludes studies that do not focus on privacy from a GDPR perspective. Similarly, studies that do not directly address the sharing and processing of IoT are excluded from the review of related works. The related works are grouped into two categories: (i) semantic-based studies, which utilize semantic technology (Section 3.1), and (ii) non-semantic technological studies (Section 3.2), which do not utilize semantic technology in their work. The other section, Section 3.3, provides a summary of the related works.

3.1. Semantic technology-based study

Rubí et al. [24] focuses on environmental data sharing in smart cities. The work uses the semantic technology, as in our study, to enable

interoperability and also takes into account the changing computing landscape, such as fog computing. The work of Rubí et al. [24], however, does not focus on other aspects, such as ensuring data veracity and GDPR. Loukil et al. [25] present a privacy-preserving blockchain-based platform, PATRIoT, for IoT data sharing. The work is GDPR-focused and makes use of semantic technology, specifically the LIoPY ontology [26], which models the privacy requirements. Moreover, the work takes into consideration the limited computing capability of IoT devices. The work, however, does not consider other aspects such as interoperability and accuracy (or veracity) of shared data, which are very important in sectors such as healthcare. Strassner et al. [27] proposed a semantic technology based architecture for interoperable IoT data sharing and also considered fog computing. However, as highlighted by Strassner et al. [27], the implementation and performance evaluation are left for future work. Reda et al. [28], likewise, conducted research on the integration and sharing of heterogeneous healthcare wearable data using semantic technology. Reda et al. [28] developed a web portal for integrating, sharing and analyzing IoT data. The proposed approach has been verified using IoT data collected in RDF (Resource Description Framework)[2] format. Reda et al. [28], similar to our study, also applied the SWRL (Semantic Web Rule Language) rule to detect the vital signs of health conditions like a cardiac event. The other aspects of the data, such as veracity, were not the focus of the work. Zappatore et al. [29] conducted a study using semantic technology for IoT in healthcare (or wellness). In their work, Zappatore et al. [29] proposed a new ontology, the FitBit API ontology, for interoperability. Moreover, it also proposed an architecture platform named App4Health and considered the edge/fog tier. However, the interoperability layer only exists at the cloud tier (or in the cloud), and further implementation is planned as a future work.

3.2. Non-semantic technological study

Makhdoom et al. [30] present a privacy-preserving and blockchain-based framework for IoT data sharing. The work by the use of the blockchain addresses the security issue and also covers GDPR requirements such as data sharing based on consent. This study, like ours, only stores the transaction hash in the blockchain. However, the study does not address the interoperability challenge and also does not provide analytics capability as in our work. Moreover, integration of fog (or fog nodes) is left as a future work.

Abdullah et al. [31] propose a privacy-aware framework, the PRISED tangle, for IoT data sharing in healthcare. The work of Abdullah et al. [31] uses IOTA Tangle, which is a zero-fee, zero-miner, zero-block distributed ledger technology (DLT) for the IoT [32]. The IOTA Tangle uses a new data structure based on directed acyclic graphs. Moreover, the authors have also considered GDPR in their framework, enabling GDPR-compliant health data sharing. With the use of the DLT, the authors have addressed privacy issues. However, issues such as interoperability and data quality remain unresolved. Bai et al. [33], similar to the previous studies, proposed a framework for the sharing and storage of healthcare data using blockchain. However, unlike previous work, the framework of Bai et al. [33] supports the GDPR's "right to be forgotten" rule and uses IPFS[3] (the InterPlanetary File System) for data storage. The framework, as the authors point out, is computationally intensive, necessitating the use of systems with greater computational capacity, and thus is unsuitable for the edge/fog scenario. Additionally, other aspects of data, such as quality, have not been considered.

Similar to Makhdoom et al. [30], Alamri et al. [34] present a framework to enable interoperable and GDPR-compliant IoT-based personal health record data sharing using blockchain. The work considers the privacy aspect, as in our study from the GDPR perspective, and also interoperability following HL7 FHIR.[4] However, the work does not ensure data veracity, and also, implementation is left as future work. Tsiouris et al. [35] propose an interoperable telehealth platform that enables the sharing of IoT data with a remote cloud for data storage and analytics. As with Alamri et al. [34], Tsiouris et al. [35] also relies on HL7 FHIR[4] standards for interoperability. The edge unit, which constitutes one of the key component of Tsiouris et al.'s [35] work, is responsible for managing IoT devices and tasks and enabling data exchange with the cloud. However, the work does not deal with issues such as data veracity (or accuracy) and privacy, which are even more critical in the health sector. Additionally, interoperability can be achieved only after data is processed in the remote cloud, as the cloud is the repository for the core interoperable technologies. Halim et al. [36] proposed a general framework for interoperable IoT data sharing. The proposed framework stores the metadata information, such as ownership, in a SQL (Structured Query Language) database and the IoT data in a NoSQL database. The framework consists of three key components: a device management layer for the abstraction of IoT devices, a data management layer for handling data, and a service layer that provides services to applications such as access to data. However, as with other studies, data veracity and quality aspects are not the focus.

Poojara et al. [37] propose a serverless data pipeline for IoT data sharing in the fog and cloud computing. The work of Poojara et al. [37] also evaluates the trade-offs of the proposed serverless data pipeline for different types of fog computing workloads, such as Anenas [38] and custom video processing. However, the work does not consider the interoperability and legal regulations such as GDPR.

3.3. Summary

In short, the review highlights that related works are devoted to specific aspects of IoT data sharing requirements such as interoperability but lack coverage of all the critical aspects: privacy, accuracy (or data veracity), and data quality, which are required for end-to-end IoT data sharing. Table 1 shows a comparison of our work to the state-of-the-art works, highlighting the current research gaps. For the comparison, we have considered the following criteria: (i) privacy from a GDPR perspective; (ii) interoperability; (ii) data quality; (iv) data veracity; (v) trust metric (or score); (vi) analytics operation indicated by analytics; (vii) considerations for the edge/fog scenario; and (viii) finally, the performance evaluation. The data quality in Table 1 reflects both the evaluation of the context information and the data themselves. In the case where a study covers either one of these data quality aspects, they are indicated with a check mark. Similarly, the analytics in Table 1 represent if the study has implemented any kinds of analytics (or reasoning) operations to support real-time decision making with an edge-and-fog scenario. The check mark (✓) indicates that the compared criteria were incorporated into the study's design, implementation, or both. The cross (×), on the other hand, indicates that either the compared criteria were not considered or there was no information in the study. As can be observed from Table 1, our study covers the needed aspects of data, such as quality, veracity, and privacy, that are required in an end-to-end data sharing scenario, thereby demonstrating the benefits of our work. Furthermore, several works on IoT privacy have been conducted, such as the one by Liu et al. [39], which do not address GDPR and have not been considered in our related works. However, such works can be regarded as relevant for future work as they can be integrated into our framework, which already enables GDPR-compliant data sharing and processing.

In light of the existing research challenges, we present our work which enables privacy-aware IoT data sharing and processing while

[2] https://www.w3.org/RDF/.
[3] https://ipfs.tech/.
[4] http://hl7.org/fhir/.

Table 1
Comparison with state-of-the-art.

Study	Privacy	Interoperability	Data quality	Data veracity	Trust metric	Analytics	Edge/Fog	Performance evaluation
Rubí et al. (2021) [24]	×	✓	×	×	×	✓	×/✓	✓
Loukil et al. (2020) [25]	✓	×	×	×	×	×	×/×	✓
Strassner et al. (2016) [27]	×	✓	×	×	×	×	×/✓	×
Reda et al. (2022) [28]	×	✓	✓	×	×	×	×/×	×
Zappatore et al. (2023) [29]	×	✓	×	×	×	✓	✓/✓	×
Makhdoom et al. (2020) [30]	✓	×	×	×	×	×	×/×	✓
Abdullah et al. (2022) [31]	✓	×	×	×	×	✓	×/×	✓
Bai et al. (2022) [33]	✓	×	×	×	×	×	×/×	✓
Alamri et al. (2021) [34]	✓	✓	×	×	×	×	×/×	×
Tsiouris et al. (2020) [35]	×	✓	×	×	×	×	✓/×	✓
Halim et al. (2022) [36]	×	✓	✓	×	×	×	✓/×	✓
Poojara et al. (2022) [37]	×	×	×	×	×	✓	✓/✓	✓
Our study	✓	✓	✓	✓	✓	✓	✓/✓	✓

simultaneously providing interoperability and assures both quality and accuracy (or veracity). In addition to privacy, interoperability, accuracy, and quality aspects, our work also considers the intelligence at the edge or fog levels to support real-time (or near real-time) decision making. For example, the increasing use of IoT for monitoring, such as in power plants, necessitates real-time decision making. This can be accomplished by the use of edge or fog computing. However, the reviewed studies fail to take into account this essential factor that our work does.

4. Methodology

This section provides details about the methodology that is adopted for ontology modeling and the trust metric (also referred to as the trust score or trustworthiness score). The methodology that is employed in our work is applicable in other contexts. Furthermore, the ontology enables the integration of heterogeneous data sources, such as in the case of smart cities [40]. Moreover, this section also provides an overview of the functional and non-functional requirements. The preliminaries of the concepts are available as supplementary material (see Appendix A).

4.1. Ontology modeling

The ontology is the core of our systems, as it facilitates interoperability and further enables reasoning capabilities. We followed the standard ontology modeling practice [41], wherein the first step is to define the scope of the ontology (or ontology requirements), such as what the ontology should model (or cover). Since this research is concerned with the sharing of IoT data, the ontology's scope should be established in a similar manner. The scope of our ontology is outlined below. The ontology as well as the source code can be accessed at [42].

1. The ontology should be able to represent sensor information and observation results, as well as context such as units of measurement describing observation results.
2. Based on what the sensors see, the ontology should be able to support reasoning (or analytics) operations, like creating an alert in our case.

According to our scope (i.e., ontology requirements), the ontology must be able to answer the competency questions (CQs) listed below. These CQs are derived from our proposed framework, which is aimed at addressing the challenges of IoT data sharing, such as data quality issues.

CQ 1. What is making an observation?
CQ 2. What is the sensor observing, such as temperature (i.e., observed property), and in what units?
CQ 3. When was the observation made?
CQ 4. How can we ensure the reliability and accuracy (i.e., data veracity) of data?
CQ 5. What type of edge reasoning is carried out?

Fig. 2 shows the ontology used in our study that was developed following our CQs. The ontology used in our study is small, i.e., it only includes a limited number of concepts pertaining to sensors and their observation. Our study's small ontology is due to the different focus of our work, which is to demonstrate the use of ontology to address challenges such as interoperability, as opposed to ontology engineering that focuses on creating large ontologies. However, the methodology adopted for our ontology modeling can be used to model larger ontologies in a collaborative setting or extend the scope of the ontology used in our study. The ontology used in this study reuses the concepts from the two different ontologies, namely SOSA [43] and the OM (Ontology of units of Measure) [44]. We consider SOSA [43] ontology compared to other ontologies like DogOnt [45] due to its wide adoption. In Fig. 2, the reused classes and properties from the OM and SOSA ontologies are denoted by the prefixes *om* and *sosa*, respectively. The prefix *sricats* shows that the concepts are unique to this study.

With CQ 1, CQ 2, and CQ 3, the ontology captures information about the sensor observation result and also context information describing the observation, such as units in which observation was made and the time when observation was made. The *sosa:Sensor* class and property *sosa:madeBySensor* provides an answer to CQ1, while *sosa:Observation* class and *sosa:-resultTime* property provides an answer to CQ3. In a similar manner, classes *sosa:Observation*, *sosa:ObservableProperty* and *om:Unit*, as well as their respective properties *om:hasUnit*, *sosa:-hasSimpleResult*, and *sosa:-observedProperty* provides an answer to our CQ 2. Moreover, the ontology can be easily expanded to include additional context information, such as location, based on the requirements. In addition, this incorporation of context information into the ontology enables us to validate using SHACL (Shapes constraint language) as part of the data quality, which is one of the limitations highlighted by [5] and one of our contributions.

The properties *sricats:hasTrustabilityScore* and *sricats: hasBlockchainHash* provides an answer to our CQ 4 and the property *sricats: hasEdgeReasoningType* as well as the classes *sricats:TemperatureAlert* and *sricats:HumidityAlert* answers CQ 5. The *sricats:TemperatureAlert* and *sricats:HumidityAlert* are classes representing alert operations specific to the humidity and temperature observations, respectively. The reason for this separation is to make concepts explicit and follows the ontology definition. These concepts, namely *sricats:TemperatureAlert* and *sricats:HumidityAlert*, can be thought of as target labels in the case of machine learning and are used in SWRL reasoning to perform the classification task. For instance, we wish to predict, based on the temperature observation value, whether or not the temperature alert should be triggered. The*sricats:-hasEdgeReasoningType* provides

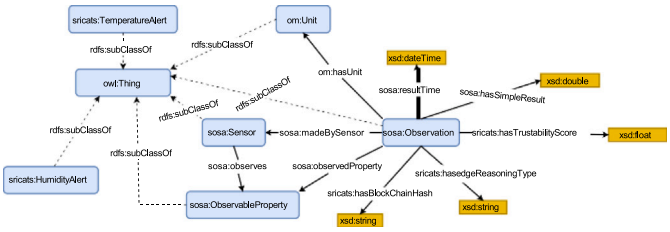

Fig. 2. Ontology used in our study.

information on the types of reasoning to perform, such as temperature or humidity in our case.

The prefix *rdfs* in Fig. 2 represents the RDF schema,[5] which provides a vocabulary for RDF. *sricats:hasBlockchainHash*, on the other hand, contains the blockchain transaction hash value, which is subsequently used to verify data veracity (or accuracy), and *sricats: hasTrustabilityScore* stores the trustworthiness score.

4.2. Trust metric

Eq. (1) illustrates the weighted trust metric (or score) that we employ in our research. The trust score (or metric) is a function of the trust factor F and its weight W. The trust score ranges from 0 to 1, with higher numbers indicating greater trustworthiness. The trust factor F defines the criteria for trust. It can be either 0 or 1, indicating whether the trust criteria have been met. W, which sums to 1, represents the weight that has been assigned to each of the established trust criteria. This weighted approach accounts for the subjectivity of trust by permitting individuals to assign weights to each of the defined trust criteria based on their level of trust.

$$trust\_score\,(t_s) = \sum_{i=1}^{n} F_i \times W_i,\ where\ F_i \in \{0,1\},\ 0 \le W_i \le 1\ and\ \sum W_i = 1 \tag{1}$$

In our study, we considered three social trust criteria (or factors): manufacturer, location where IoT is deployed, and deployment by, i.e., who deployed the IoT. The weights assigned were 0.5, 0.3, and 0.2 for manufacturer, location, and deployment, respectively. The considerations for these trust factors derive from [46,47]. For instance, if an IoT device manufactured by Texas Instruments is deployed, the level of trust would be high. This is because of their reputation, and studies [46,47] have already demonstrated the relation between social factors such as reputation and trust in the case of the IoT. However, these criteria can be easily expanded to incorporate additional criteria; doing so is a trivial task.

4.3. Functional and non-functional requirements

In this section, we describe the system's functional and non-functional requirements. These requirements are derived from our research questions and are consistent with other research. For instance, Gupta [48] has also highlighted the non-functional requirement of performance efficiency, one of the key challenges in edge (or edge computing) due to its resource-constrained nature. The below-described functional and non-functional requirements serve as the premise for designing the proposed system.

Functional requirements:

1. Upon receiving (or reading) the sensor data, the system must check for consent before further processing.
2. Given the successful consent check, the system should compute the trust score, generate the hash of the data for the data veracity check, and store the generated hash in the blockchain.
3. After successfully calculating the trust score and storing the hash in the blockchain (or blockchain hash) the system should convert the sensor data, the trust score, and the blockchain hash into the KG representation according to the ontology.
4. Upon data migration to a new location, the system should be able to reconstruct the hash from the KG and conduct a data veracity check. In addition, the system should be able to perform data quality tests using SHACL after data veracity checks have been completed successfully.

[5] https://www.w3.org/TR/rdf-schema/.

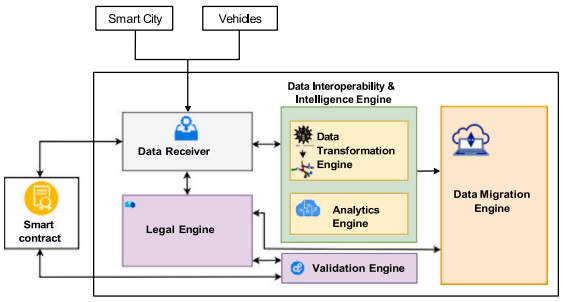

Fig. 3. Proposed framework architecture.

Non-functional requirements:

1. The proposed system should be resource-efficient, i.e., suitable for deployment on resource-constrained IoT devices, with minimal CPU and memory usage.
2. To support scalability, the system should permit the disintegration of its components similar to microservices architecture.
3. The system should be lawfully compliant; for instance, data processing should only occur with explicit consent, and the use of blockchain should not violate GDPR.
4. The system should support real-time (or near real-time) decision making.
5. The system should support interoperability and ensure the data's accuracy and trustworthiness.

5. Proposed system

This section provides a conceptual high-level architecture description designed after reviewing the existing works. Fig. 3 shows the architecture of the proposed system (or framework). Additionally, the design of the proposed system is governed by the functional and non-functional requirements outlined in Section 4.3. The following subsections provide detailed information about the proposed system's components, consistent with other studies [21,22,24,25,36].

5.1. Data source

The data source comprises various sources that generate data through IoT devices. These sources include smart cities, smart vehicles, smart homes, wearable health devices, and hospitals, such as using environmental sensors in smart cities to monitor noise levels, air quality, and pollution and smart inhalers in hospitals to monitor patients.

5.2. Smart contract

The smart contract in the framework represents a blockchain smart contract. In our case, it represents the Ethereum blockchain smart contract. The smart contract in our study contains the smart contract definitions (i.e., code), such as to store the hash. The reason for using

the blockchain to store hashes is because generated hashes can be compromised by exploiting flaws [49]. The storage of hashes in the blockchain ensures that the hashes are tamper-proof and can be relied upon while performing a data veracity check.

5.3. Data receiver

The data receiver is the first component to interact with the data from the sensor and is the one upon which the other components of the proposed systems depend, such as the analytics engine, to perform tasks such as data analytics in order to enable intelligence. The received sensor data is processed, such as by enriching it with additional context information like observation units, prior to being transformed into the KG representation for interoperability. Algorithm 1 shows the steps for data processing by the data receiver component.

Algorithm 1: Algorithm for sensor data processing

```
Input: Sensor observation value and data subject consent
Output: Data in KG representation
sensorSetup ← Sensor configuration information such as name and trust factors;
if check_consent = True then
    data ← get_sensor_observation_value();
    trust_score ← compute_trust_score(sensorSetup);
    data_hash ← generate_hash(data, trust_score, timestamp);
    blockchain_hash ← store_data_in_blockchain(data_hash);
    data_in_kg_format ← transform_to_kg(data, blockchain_hash, trust_score,
      sensorSetup);
    return data_in_kg_format;
else
    No processing;
end
```

As the aim is also to enable GDPR-compliant data sharing, first the consent is checked to ensure that there is appropriate consent from data subject (or natural person) to process sensor data. For this data receiver component interacts with the legal engine of the framework. After that, the sensor data is read, and the trust score is computed (see Section 4.2). The trust score calculation in Algorithm 1 is indicated by the function $compute\_trust\_score(sensorSetup)$. The $sensorSetup$ contains information about the sensors and the trust factors (or criteria). Following the computation of the trust score, a hash is generated, which is later used for data veracity purposes. The hash is generated using a timestamp (date and time) value together with the trust score and the actual data. The computation of the hash is indicated by $generate\_hash(data, trust\_score, timestamp)$. sha256 is used for hash generation. The generated hash, represented by $data\_hash$ variable in Algorithm 1, is then stored in a blockchain smart contract, ensuring the integrity of the hash because of the immutability of the blockchain. The blockchain hash (i.e., $blockchain\_hash$ in Algorithm 1) is a hash of a blockchain transaction that was generated while storing the data hash in blockchain, which in our case is the Ethereum. This is especially important because it enables data sharing in untrusted environments with the assurance that data veracity can be verified. The reason for only storing hashes instead of complete data in the blockchain is because of the contradiction of the blockchain with the GDPR's right to erasure (Article 17), as data stored in the blockchain cannot be deleted. Finally, the data is transformed into a KG representation, using the data, blockchain hash, trust score, and sensor configuration information, and by interacting with the transformation engine. $transform\_to\_kg(data, blockchain\_hash, trust\_score, sensor-Setup)$ in Algorithm 1 represents this transformation stage. In the absence of consent, no processing, not even reading the sensor data, is performed.

5.4. Data transformation engine

The data transformation engine is one of the core components, its tasks include the conversion of raw data to their respective semantic representations, namely, KG, thereby enabling interoperability. The interoperability occurs through the use of the ontology that defines concepts with a shared meaning (see evaluation Section 7.2.4) [50,51]. The following steps summarize the data transformation process.

1. The first step is to define the namespaces[6] by reusing the existing ontology where possible.
2. The subsequent step involves extracting the relevant information from the input data.
3. The third (or final) step is to perform a mapping of the extracted information from the input data as per the concepts (i.e., class, object properties, and data properties) defined in an ontology. The descriptions of the ontology used in our study are presented in Section 4.1.

Additionally, the data transformation engine converts the generated KG into a format compliant with SPARQL[7] for data migration. SPARQL is a query language for KG (or RDF graphs).

5.5. Legal engine

The legal engine is responsible for checking if there is valid consent for the requested data processing activity, thereby ensuring that the processing is GDPR-compliant. The steps involved in checking the consent are shown in Algorithm 1. First, consent is obtained from the data subject (or natural person) and is checked against the data processing operations that is intended to be carried out. This is to ensure that the data processing activities adhere to the consent provided. The data processing information is obtained from other components of the framework at different stages, such as during initial sensor data processing and during validation and analytics operations. For example, one can deploy multiple sensors at home but may consent to share or process certain sensor data, and that too for a specific purpose. Following the check, the final status represented by true or false is returned. The status indicates that everything is fine and data processing activities can be carried out, while the false status indicates that there is no consent. "No consent" here could mean either no consent at all or not for the requested processing activity or sensor. In Algorithm 1 $processing\_information()$ and $obtain\_consent\_from\_data\_subject()$ are used to obtain the data processing and consent information, respectively.

Algorithm 2: Algorithm for checking consent

```
Input: Data and consent from data subject
Output: Status indicating whether or not to continue data processing activities
consent ← obtain_consent_from_data_subject();
dataProcessing ← processing_information();
if check(consent, dataProcessing) = True then
    return status True;
else
    return status False;
end
```

5.6. Validation engine

The validation engine performs the data validation tasks to ensure the following: (i) data accuracy (or veracity); and (ii) data quality. The accuracy (or veracity) of the data ensures that it has not been altered and that the data at the destination is the same as at the source where it originated. The data quality, on the other hand, ensures that the shared sensor data meets the predefined quality criteria. For the quality criteria, the following were considered: (i) the existence of the properties sosa:observedProperty, sosa:observes, and sosa:madeBySensor with type IRI; (ii) the existence of the sosa:resultTime with data type datetime; (iii) the presence of the blockchain with a minimum length of 60 with data type string; (iv) the presence of the sensor observation result

[6] https://www.w3.org/TR/2004/REC-owl-guide-20040210/#Namespaces.
[7] https://www.w3.org/TR/sparql11-overview/.

of type double and the range between 0–100; and (v) the occurrence of the trustability score of type float. These quality criteria can be extended easily to include additional context and data sources by defining them in the SHACL, and doing so is trivial. Algorithm 3 outlines the steps to perform the validation.

Algorithm 3: Algorithm for performing validation

```
Input: Data in KG format
Output: Status and validated data in KG in the case of successful validation in a
        dictionary format
1  if check_consent = True then
2      sensor_observation ← get_observation_result(input KG);
3      trust_score ← get_trust_score(input KG);
4      timestamp ← get_timestamp(input KG);
5      constructed_hash ← reconstruct_hash_from_data(sensor_observation, trust_score,
         timestamp);
6      blockchain_hash ← get_stored_hash_from_blockchain(input KG);
7      hasDataVeracity ← check_data_veracity(blockchain_hash, constructed_hash);
8      if hasDataVeracity = True then
9          shacl_rules ← get_shacl_rules(input KG);
10         meetsDefinedDataQuality ←validate_for_quality(transformed_kg,
             shacl_rules);
11         if meetsDefinedDataQuality = True then
12             return status True along with RDF representation of KG;
13         else
14             return status False;
15         end
16     else
17         return status False;
18     end
19 else
20     No processing;
21 end
```

As with the case of sensor data processing (see Section 5.3), first the consent check is performed by interacting with the legal engine component of the framework. This method of checking consent prior to each processing activity is intended to enable GDPR-compliant data sharing or processing at the granular level. After the successful consent checks, the relevant information from the data, i.e., KG, is extracted. This includes getting the sensor observation value, the trust score, and the time stamp information, as indicated by steps 2–4 in Algorithm 3. After that, using the sensor observation value ($sosa : hasSimpleResult$ value in KG), timestamp ($sosa : resultTime$ value in KG), and trust score ($sricats : hasTrustabilityScore$ value in KG), the hash value is reconstructed in order to verify its veracity. Following the reconstruction of the hash, the original hash stored in the blockchain is retrieved using the transaction hash information from KG (i.e., $sricats : hasBlockchainHash$).

In Algorithm 3, step 5 is the step of reconstructing the hash, and step 6 is the step of getting the original hash from the blockchain. Veracity is determined by comparing the original and recomputed (or reconstructed) hashes. This aids in the detection of tampered data, which is crucial in industries such as healthcare. Additionally, the storage of the original hash in the blockchain guarantees the tamper-proof nature of the stored hash value due to the immutability of the blockchain, thereby further guaranteeing the veracity of the shared data.

Steps 7–8 of the algorithm show the data veracity check. The data quality test is conducted after the data veracity test has been passed. For a quality test, we begin by getting the SHACL rules ($get\_shacl\_rules(input\ KG)$), which contain the predefined quality criteria based on the sensor data observation, such as temperature and humidity. Once the necessary SHACL rules have been obtained, the KG is validated against the SHACL rules ($validate\_for\_quality(input KG, shacl rules)$) to determine if it meets the quality requirements. The evaluation of quality also test context information, such as the observation units, which is one of our contributions. If the quality test is passed, both the status and the KG are returned, whereas in the case of failure, only the status $False$ is returned, indicating that the defined quality criteria is not met.

5.7. Analytics engine

The analytics engine is the intelligence-enabling component of the proposed system and provides analytics capabilities to facilitate data-driven decision making. For instance, activating the cooling system (or taking other appropriate action) in the event of a high temperature in the data center. This is because a high temperature can cause performance degradation and also system failures that result in outages. Moreover, the analytics engine facilitates the data-driven decision making capabilities at various layers. For example, in the fog and edge layers to enable latency-aware decision making. The analytics engine, however, can also be extended to the cloud to facilitate data-driven decision making in situations where real-time requirements are not necessary (or latency is not a problem). In our study, to facilitate data-driven decision making, the analytics engine implements KG-based reasoning techniques like SWRL reasoning. The SWRL reasoning is used to perform the classification task, i.e., classifying *sricats:TemperatureAlert* and *sricats:HumidityAlert* based on the observed values of temperature and humidity. The capability of the analytics engine can easily be extended further beyond SWRL reasoning to machine learning techniques, taking advantage of the KG for improved prediction, as shown by studies such as [52]. Similar to the initial sensor data processing and validation of the veracity of the data, the consent is checked prior to performing analytics operations.

5.8. Data migration engine

The data migration engine, as the name suggests, performs data migration (or transfer) operations, an essential operation in data sharing. The data migration engine performs the migration (or transfer) of the data from source to destination. The IoT sensor represents the source, and the remote storage, a graph database, represents the final destination.

6. Implementation

This section provides information about the system setup that is used for implementation and experimentation. In addition, this section also provides details on the implementation. Section 6.1 provides system setup information and Section 6.2 describes the implementation in detail.

6.1. System setup

In this section, we describe the libraries used to implement the proposed system as well as the experimental system. Table 2 summarizes the list of libraries used for the implementation. Python[8] is used for the implementation due to its popularity in the data science community and its usability. The libraries RDFLib,[9] pySHACL[10] and Owlready2[11] were used to deal with the semantic technology, namely, KG and ontology. The library RDFlib is used for parsing and serialization of the KG represented in RDF format, while pySHACL is used to perform the validation and Owlready2 for reasoning. The library Pika[12] is used to interact with RabbitMQ,[13] while PyYAML[14] is used to manage YAML[15] configuration files and Adafruit-DHT[16] is used to interact with the DHT11 sensor. Docker[17] is used for deployment of systems such as RabbitMQ. Solidity and py-solc-x[18] are used to

[8] https://www.python.org.
[9] https://rdflib.readthedocs.io/en/stable/.
[10] https://github.com/RDFLib/pySHACL.
[11] https://owlready2.readthedocs.io/en/v0.37/.
[12] https://pika.readthedocs.io/en/stable/intro.html.
[13] https://www.rabbitmq.com.
[14] https://pyyaml.org/wiki/PyYAML.
[15] https://yaml.org.
[16] https://pypi.org/project/Adafruit-DHT/.
[17] https://www.docker.com.
[18] https://solcx.readthedocs.io/en/latest/.

Table 2
Information about the libraries that were used in the implementation.

Libraries/Software	Version
Python	3.8
RDFLib	6.1.1
PyYAML	6.0
pySHACL	0.19.0
Pika	1.2.1
Owlready2	0.30
Adafruit-DHT	1.4.0
Docker	20.X
Web3	5.31.1
Ganache-cli	2.13.2
Solidity	0.8.12
py-solc-x	1.1.1

Table 3
Information about the system and its configuration used in the experiment.

System information	System configuration
Raspberry Pi 3 B+ (or Edge)	1 GB RAM, 4 core Cortex-A53 processor, 16 GB storage, Raspbian GNU/Linux 10 (Buster) OS
MacBook Pro (or Fog)	16 GB RAM, 2,8 core 3 GHz Quad-Core Intel Core i7 processor, 500 GB storage, macOS Monterey version 12.3.1
RabbitMQ messaging server and Ganache-CLI	64 GB RAM, 16 core Intel Core Processor (Broadwell), Ubuntu 20.04 LTS (GNU/Linux 5.4.0-80-generic x86_64) OS, 32 GB with additional 2 TB shared storage
GraphDB server	64 GB RAM, 16 core Intel Core i9-9900K processor, 1 TB storage, Debian GNU/Linux 10 (buster) OS

implement the Ethereum smart contract. Ganache-cli[19] is the Ganache command line interface. Ganache is a personal blockchain for Ethereum that enables the safe and deterministic development, deployment, and testing of decentralized applications.[20]

Table 3 summarizes the configuration of the system used in our experiment. In our experiment, the Raspberry Pi 3 B+ serves as the edge (or edge device), emulating a real-world scenario in which edge devices are typically those with the lowest system capabilities (or are usually the resource-constrained devices). The fog (or the fog layer), which is one step up in the hierarchy, contains the systems with higher configurations than that of the edge. In our case, the system MacBook Pro represents the fog, the configuration of which is shown in Table 3. Moreover, Table 3 also shows the system configuration of the servers used for GraphDB and RabbitMQ.

6.2. Implementation

In this section, we provide an overview of the implementation of the proposed system. The source code for the implementation is available at [42]. All of the implementations of the framework components are performed using Python.

6.2.1. Smart contract

The Ethereum smart contract used in our study is implemented in Solidity. Solidity is a high-level language for implementing smart contracts. However, as our framework is implemented in Python, our implementation of the smart contract uses py-solc-x, a Python wrapper for the Solidity compiler. The smart contract is implemented in Solidity version 0.8.12 and deployed using Ganache-cli.

[19] https://docs.nethereum.com/en/latest/ethereum-and-clients/ganache-cli/.

[20] https://trufflesuite.com/docs/ganache/.

```
@prefix om: <http://www.ontology-of-units-of-measure.org/resource/om-2/> .
@prefix sosa: <http://www.w3.org/ns/sosa/> .
@prefix sricats: <http://www.abc.com/sricats/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://www.w3.org/ns/sosa/Observation/DHT11_202212025703> a sosa:Observation ;
    om:hasUnit om:percent ;
    sricats:hasBlockChainHash "0xcbb35fbc905aa6aeceaeebec596bc76937d6a0cd1e04ff3aa970adc13899b076" ;
    sricats:hasTrustabilityScore "0.8"^^xsd:float ;
    sosa:hasSimpleResult 2.8e+01 ;
    sosa:madeBySensor <http://www.w3.org/ns/sosa/Sensor/DHT11> ;
    sosa:observedProperty <http://www.w3.org/ns/sosa/observedProperty/STI_W201_humidity> ;
    sosa:resultTime "2022-12-02T20:57:03"^^xsd:dateTime .

<http://www.w3.org/ns/sosa/Sensor/DHT11> a sosa:Sensor ;
    sosa:observes <http://www.w3.org/ns/sosa/Observation/DHT11_202212025703> .
```

Fig. 4. Generated KG after performing a data transformation operation on the humidity observation from DHT11 sensor.

6.2.2. Data receiver

The data receiver components implement Algorithm 1, outlined in Section 5.3, in order to read and process sensor data. In scenarios such as fog–edge, the data receiver also implements the functionality to read data from the messaging server. In our research, we utilized the DHT11 sensor. Thus, the data receiver reads the DHT11 sensor data. In the fog–edge scenario, the DHT11 sensor data represented in KG format at the fog layer is retrieved from the publish–subscribe model messaging server RabbitMQ via the topic *secure_interoperable_data_sharing*. The messaging server, RabbitMQ, is implemented using Docker on four cluster nodes to provide fault tolerance. The Docker configuration details are available at [42]. For the computation of the trust score, a separate rest service is implemented. It can, however, be computed without additional rest implementations. The reasons for this are that trust factors change over time and that more transparency is desired. For example, if the manufacturer improved their IoT product safety and obtained certification like ISO (International Organization for Standardization) security certification, then one may want to update the trust score computation. If the computation does not follow our approach and is done locally, updating in the case of large scale IoT is extremely inefficient. The second reason is that it enables transparency, i.e., the sharing of the factors used to calculate the trust score, which is essential because trust is subjective and different deployments may take into account different factors.

6.2.3. Data transformation engine

The implementation of the data transformation engine, following the procedures mentioned in Section 5.4, transforms the data into the KG. The KG representation of the humidity sensor data following the data transformation procedure is shown in Fig. 4. As shown in Fig. 4, the data (or KG) has been enriched with additional context information, including observation units and observed property. The KG shown in Fig. 8 is in a Turtle format. The data in KG representation is shared using JSON-LD. JSON-LD is a lightweight JSON (JavaScript Object Notation)-based format to serialize linked data.[21]

6.2.4. Legal engine

In general, the legal engine implementation consists of consent verification, as described in Section 5.5. In our investigation, however, we simulated consent checking by passing the boolean value true to represent the consent scenario and false to represent the absence of consent or invalid consent scenarios. The reasons are as follows: (i) there are already studies, such as [22], that perform consent checking and other complex consent-related operations; and (ii) integrating such systems is trivial. Fig. 5 shows the response when there is no consent for the requested processing, indicating that the data processing (or sharing) action cannot be executed.

[21] https://www.w3.org/TR/json-ld11/

```
{"message":"Unable to process due to a lack of consent for
            the requested operation."}
```

Fig. 5. Response when no consent is available for request processing.

6.2.5. Validation engine

The validation engine implementation is the realization of the validation algorithm, Algorithm 3, described in Section 5.6. In our implementation, we defined the following quality criteria for both context and the actual sensor data as SHACL rules: (i) the *sosa:hasSimpleResult* must have a value of type *xsd:double* with a maximum value of 100, (ii) *sricats:hasBlockChainHash* must have a value of type *xsd:string* with the minimum length of 60 characters, (iii) the *om:hasUnit* must have a value of a defined type based on sensor (e.g., *om:degreeCelsius* for temperature sensor and *om:percent* for relative humidity), (iv) *sricats:hasTrustabilityScore* must have the value of type float and (v) the property *sosa:madeBySensor*, *sosa:observes* should be of type IRI and must have a value. Fig. 6 shows the SHACL validation result. Evidently, both validations (see Fig. 6(a) and Fig. 6(b)) fail the predetermined quality criterion, resulting in the return of the false status. In the event of a successful validation, the KG and the status true are returned.

6.2.6. Analytics engine

As discussed in Section 5.7, the analytics engine facilitates data-driven decision making. The analytics engine in our study employs SWRL reasoning using Pellet[22] reasoner to facilitate data-driven decision making. Fig. 7 illustrates the SWRL rules utilized in this study to demonstrate data-driven decision making. Our study's data-driven task is a classification task (i.e., whether or not to classify *TemperatureAlert* and *HumidityAlert*) based on observations of humidity and temperature. The *?result* in the SWRL rules contains sensor data, whereas the *?reasoningType* gives information regarding the type of reasoning based on the observation made. For example, the *?reasoningType* for a temperature observation is the value temperature. Fig. 8 shows the snippet of edge analytics result indicating the high-temperature alert. However, if no alert condition is met, the reasoning result, i.e., alert value, is none, indicating everything is fine. When the analytics operation is performed in the fog, the results are published to the messaging server (see Section 7.1.2) and consumed at the edge.

6.2.7. Data migration engine

As stated in Section 5.8, the data migration engine migrates data from one location to another. This migration also encompasses the transfer of data to intermediary phases, such as messaging servers. The implementation consists of the following: (i) migrating the data (i.e., KG) to the messaging server from edge, which will later be used at the fog layer for further processing (see Section 7.1.2); (ii) migrating data (i.e., KG) to the remote storage; and (iii) migrating the analytics result to the edge in the case when the analytics are performed at layers such as fog (see Section 7.1.2). The publish/subscribe model is used for data and result migration when processing, such as when analytics are conducted in the fog. The topic *secure_interoperable_data_sharing* is used for the data migration. Similarly, the topic *reasoning_result_secure_interoperable_data_sharing* for the result (i.e., analytics result), which in our study is to issue a warning (or alert). SPARQL is used for migrating the KG data to the remote storage. GraphDB, which is a W3C[23] compliant enterprise-ready semantic graph database,[24] is used as the remote storage. Table 4 shows the KG in triples representation (i.e., subject, predicate and object format) in GraphDB after migration.

[22] https://www.w3.org/2001/sw/wiki/Pellet.
[23] https://www.w3.org.
[24] https://graphdb.ontotext.com.

7. Experiment and evaluation

This section describes the experiment in detail. In addition, this section provides information about the evaluations conducted. Section 7.1 discusses the experiment and Section 7.2 discusses the evaluation.

7.1. Experiment

In this section, we will describe our experimental scenario. Section 7.1.1 describes the edge experimental scenario and Section 7.1.2 describes the edge–fog experimental scenario.

7.1.1. Edge experiment scenario

Fig. 9 shows the overview of the edge experiment scenario that we considered in our study. As shown in Fig. 9, sensor data is read and processed on the edge device, performing tasks such as data transformation and edge analytics to enable edge intelligence. Below are the specifics regarding the operations performed at each step.

1. The initial step (step 1) checks for consent to ensure that there is permission to read and process the sensor data. The consent check is performed to ensure that the data sharing and processing complies with GDPR. If there is consent, then the sensor data is read and processed (step 2), i.e., transformed into the KG representation. In our case, we utilize the DHT11, a temperature-humidity sensor. If the consent check is unsuccessful, no further processing occurs.
2. Similar to the initial step, the consent is checked again (step 3) before performing other operations, such as validation, analysis, and transformation.
3. After a successful consent check at step 3, the data is validated by utilizing SHACL for quality and integrity, which is then transformed into a format compliant with SPARQL and migrated to the remote graph database (or GraphDB). This process is indicated by steps 4, 5, and 6.
4. Similar to the migration case, the analytics operation (step 7) is done after the consent check (step 3) and validation (step 4) have been completed successfully.

The data transformation and migration step and the data analysis step could be executed concurrently. In our study, we did not perform concurrent implementations as it was not the main focus of our work. However, such optimizations are important and can be regarded as future work.

7.1.2. Fog–edge experiment scenario

Fig. 10, similar to Fig. 9 (i.e., edge scenario), shows the overview of the experiment scenario that we considered in our study. In contrast to Fig. 9, which only considers the edge scenario, we considered the fog–edge scenario in this experiment. Unlike in the edge experiment scenario, where all the tasks are performed at the edge, in this scenario, the work is distributed between the fog and the edge. The descriptions of the operations performed at each step are provided below.

1. The first two steps at the edge are similar to those in the edge experiment: check consent and read and process the data, as described in Section 7.1.1. The third step, consent checking, is quite different from the case of the edge experiment, though consent checks are performed at both steps. This is because the consent check at this step in the fog–edge experiment checks to see if the data can be migrated to the messaging server. In the edge experiment (see Section 7.1.1), consent is checked to see if the data can be processed, such as to perform validation.

Fig. 6. SHACL validation. (a) SHACL validation report for validation failure as a result of the observation value falling outside of the specified range. (b) SHACL validation report for failed validation due to the KG not meeting the predefined quality criterion observation unit.

Table 4
A snapshot of the migrated KG instance to the remote storage GraphDB. *xsd*, *sricats*, *sosa*, *om*, and *rdf* are the prefixes.

Subject	Predicate	Object
sosa:Observation/DHT11_202212020002	om:hasUnit	om:percent
sosa:Observation/DHT11_202212020002	sricats:hasBlockChainHash	"0x4078c7919896566330e95bf68f9106dcb eec1ef7df19b6fda6e559347a933bd1"
sosa:Observation/DHT11_202212020002	sricats:hasTrustabilityScore	"0.8"^^xsd:float
sosa:Observation/DHT11_202212020002	rdf:type	sosa:Observation
sosa:Observation/DHT11_202212020002	sosa:hasSimpleResult	"28.0"^^xsd:double
sosa:Observation/DHT11_202212020002	sosa:madeBySensor	sosa:Sensor/DHT11
sosa:Observation/DHT11_202212020002	sosa:observedProperty	sosa:observedProperty/STI_W201_humidity
sosa:Observation/DHT11_202212020002	sosa:resultTime	"2022-12-02T21:00:02"^^xsd:dateTime

```
Observation(?observation),
hasSimpleResult(?observation, ?result),
hasedgeReasoningType(?observation, ?reasoningType),
containsIgnoreCase(?reasoningType, "temperature"),
greaterThanOrEqual(?result, 75.0),
-> TemperatureAlert(?observation)

Observation(?observation),
hasedgeReasoningType(?observation, ?reasoningType),
containsIgnoreCase(?reasoningType, "humidity"),
hasSimpleResult(?observation, ?result),
greaterThanOrEqual(?result, 65.0),
-> HumidityAlert(?observation)
```

Fig. 7. SWRL rules for temperature (top) and relative humidity (bottom) alerts used in our study.

```
{"alert": "TemperatureAlert", "observationsensorid": "DTH11",
"observedproperty": "STI_W201_temperature",
"observationid": "DTH11_20220514220973375 5"}
```

Fig. 8. Snippet of results of the edge analytics are based on the temperature observation indicating a high-temperature alert in accordance with the SWRL rule.

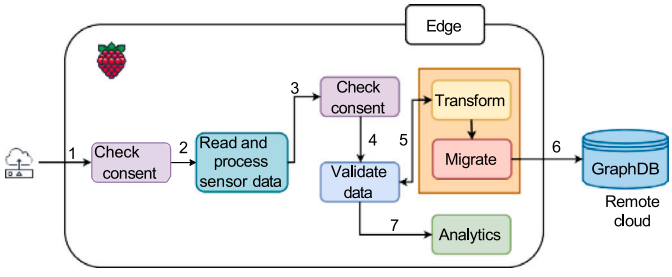

Fig. 9. High-level overview of edge experiment scenario.

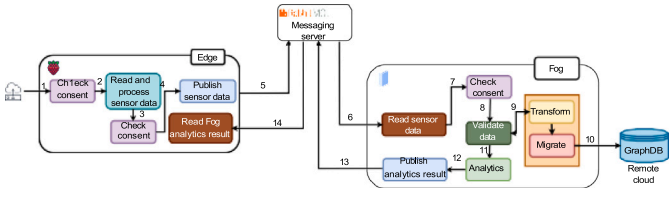

Fig. 10. High-level overview of fog–edge experiment scenario.

2. The data is migrated (steps 4 and 5) to the messaging server after a successful consent check. This data migration is done to delegate tasks, such as performing data analytics, to the fog. However, if there is no consent, then no processing, which in this case is the migration of data, is performed. The messaging server collects and holds published data (or message) in a messaging queue until it is consumed. Moreover, the messaging server, which acts as an intermediary between fog and edge, also facilitates communication and data exchange.
3. At this stage (step 6), the data from the messaging server is read (or consumed). After reading the sensor data from the messaging server, another consent check is performed (step 7). The consent check at this step is similar to the case of the edge experiment (see Section 7.1.1) consent check at step 3. This is because, similar to the case of the edge experiment, the consent check in step 7 of the fog–edge experiment scenario involves checking consent to perform tasks such as validation.
4. Following the successful consent check at step 7, operations similar to the edge experiment are performed. This included performing validation for integrity and quality (step 8), as well as transformation and migration (steps 9 and 10), and migrating the data to the remote storage, GraphDB. The additional consent check at the fog ensures that everything is compliant with the GDPR. In addition, this helps ensure that privacy will be maintained even if the consent check has been compromised at the edge.
5. As with the transformation and migration operations, the analytics operation (step 11) is also performed. However, unlike the case of the edge experiment, where the analytics results are returned instantly, in this case the result is published (or migration of analytics result) back to the messaging server from the fog. Steps 12 and 13 in Fig. 10 indicate these processes.
6. The messaging server collects and holds the published results from the fog, which is then consumed by edge (step 14) and no further processing was performed. However, some control actions, such as issuing alerts (or taking appropriate action), can be performed based on the analytics outcomes.

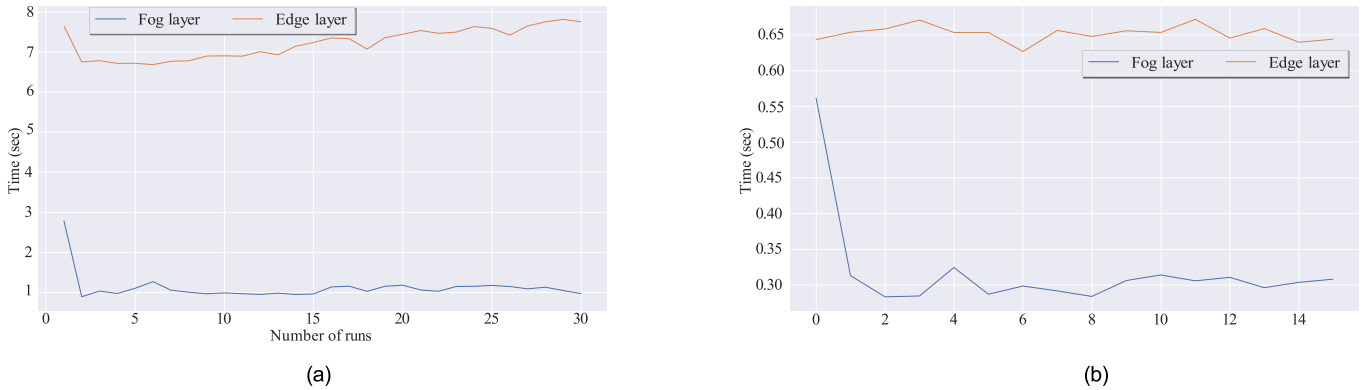

Fig. 11. Execution time. (a) Time required to execute analytics at the edge and fog layers. (b) Time required to execute data transformation and migration at the edge and fog layers.

7.2. Evaluation

In this section, we provide details about the evaluation. Section 7.2.1 provides details on the evaluation criteria and Section 7.2.2 provides details on the performance evaluation conducted following our evaluation criteria. Similarly, Section 7.2.3 provides an overview about data quality, assessment of the context information as component of data quality and data veracity evaluation, while Section 7.2.4 provides information regarding interoperability evaluation. Section 7.2.5 provides detail on the evaluation of analytics operation.

7.2.1. Evaluation criteria

We evaluate the proposed work to determine its applicability in a real-world setting and to address our research questions (or to determine whether our objective has been achieved). The following evaluation criteria were utilized to assess the proposed work:

1. The first criterion is the resource utilization. IoT, by its very nature, has limited resources, such as CPU and memory, so implementing the proposed framework should utilize the minimum (or reasonable) amount of resources. Section 7.2.2 presents the evaluation of resource utilization.
2. The second evaluation criterion focuses on accessing the proposed work's capability to support real-time (or near real-time) decision making in the edge/fog environment. It is determined by the execution time to perform the analytics operation and is presented in Section 7.2.2 and Section 7.2.5.
3. The third criterion is that the proposed system should support accurate (e.g., data veracity) and quality data sharing. The fourth criterion is that the system should support interoperability. Section 7.2.3 presents the evaluation of the third criterion. Section 7.2.4 presents an evaluation of the fourth criterion.

7.2.2. Performance evaluation

We conducted the performance evaluation to analyze the feasibility of the proposed system. We focused on the following in our performance evaluation: (i) execution time to perform the operations such as analytics, data transformation, and migration; (ii) resources such as memory and CPU (central processing unit) utilization and (iii) latency for analytics operations in the case of the fog–edge experiment. The reason for focusing on memory and CPU resource utilization other than the execution time is that the edge devices are typically resource-constrained in nature. Evaluating resource use is therefore crucial to understanding the feasibility of the proposed system in a real-world deployment.

Fig. 11 illustrates the performance evaluation in terms of the execution time. Fig. 11(a) shows the execution time required to execute analytics at the edge and fog levels, whereas Fig. 11(b) depicts the execution time required to do data transformation and migration at the edge and fog layers. Overall, the execution time for doing the analytics (see Fig. 11(a)) at the fog layer was an average of 1.1 s, whereas the execution time at the edge was an average of 7.2 s. We observe a similar tendency for doing analytics in both the fog and edge layers. For example, a high execution time for the initial reasoning is followed by a significant drop in execution time. In addition, the execution time for both fog and edge layers exhibits a constant increase with intermittent spikes. On closer inspection, particularly at the end (i.e., after 25 iterations), we can see that the execution time at the edge increases while the fog decreases. The higher execution time at the edge can be attributed to the limited availability of resources. As seen in Fig. 11(a), for example, the fog, which has more available resources than the edge, has a reduced execution time. Similar to the analytics scenario, the execution time for data transfer and migration activities (see Fig. 11(b)) follows a similar pattern, with a high execution time, in the beginning, followed by a fast decrease and gradual increase, with the occasional spike in between. Unlike the analytics scenario, the execution time exhibits a similar pattern even at the end. Data migration and transfer procedures at the edge took approximately 0.65 s and 0.32 s in the fog, respectively. Based on Figs. 11(a) and 11(b) (i.e., execution time), we can conclude that data migration and transfer procedures are less time-consuming than analytical processes. Moreover, the execution time, both for the analytics and the transformation and migration operations, also accounts for the time taken for intermediate steps, such as performing validation checks.

Figs. 12 and 13 illustrate the resource use evaluation. Similarly, Fig. 14 shows the latency in getting analytics results and the time taken for creating blockchain transactions. Figs. 12(a) and 12(b) focus on evaluating the memory utilization, while Figs. 13(a) and 13(b) focus on evaluating the CPU utilization. We only consider the edge for memory utilization for analytics and data transfer and migration operations. This is because the edge, comparatively, is much more resource-constrained than the fog. Therefore, the fog, which has more available memory than the edge, will not experience memory-related performance degradation. For CPU utilization, however, we consider both the fog and edge similar to that of execution time. This is because, unlike memory, CPU utilization can vary depending on the tasks, such as SWRL reasoning for analytics operations, and has an impact on execution time. As can be observed in Figs. 12(a) and 12(b), the memory utilization for both the data migration and transformation and analytics operations is nearly same, which is around 51 MB (megabytes). Moreover, memory usage in both instances follows a similar pattern: a sudden increase followed by a continuous increase. The sudden increase in memory indicates the start of tasks such as analytics or data processing and memory allocation. This low memory usage demonstrates the practicality of the proposed framework for resource-constrained devices such as the one in the edge layer, which is usually resource-constrained in nature.

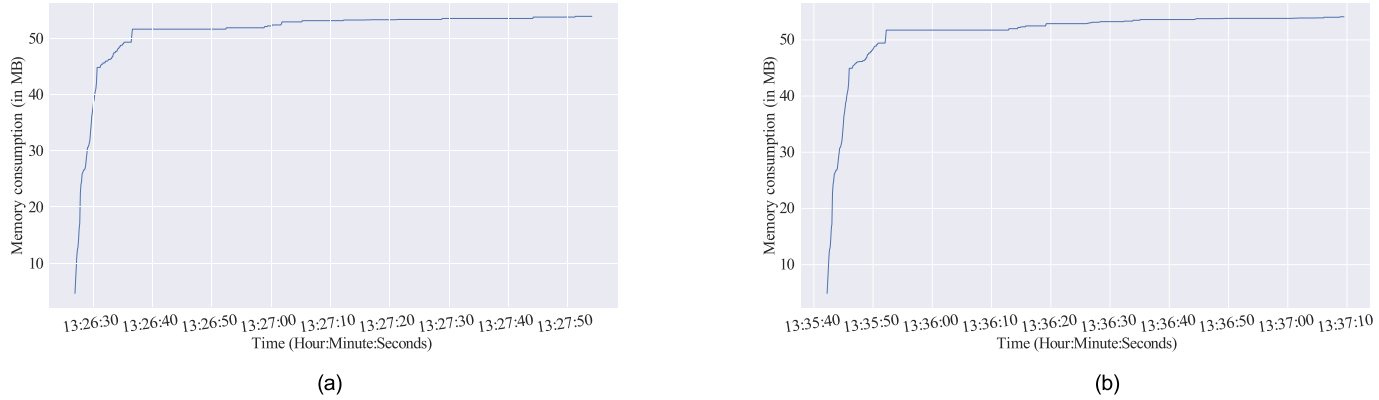

Fig. 12. Memory utilization at the edge. (a) Memory consumption to perform data transformation and migration operations at the edge layer. (b) Memory consumption to perform analytics at the edge layer.

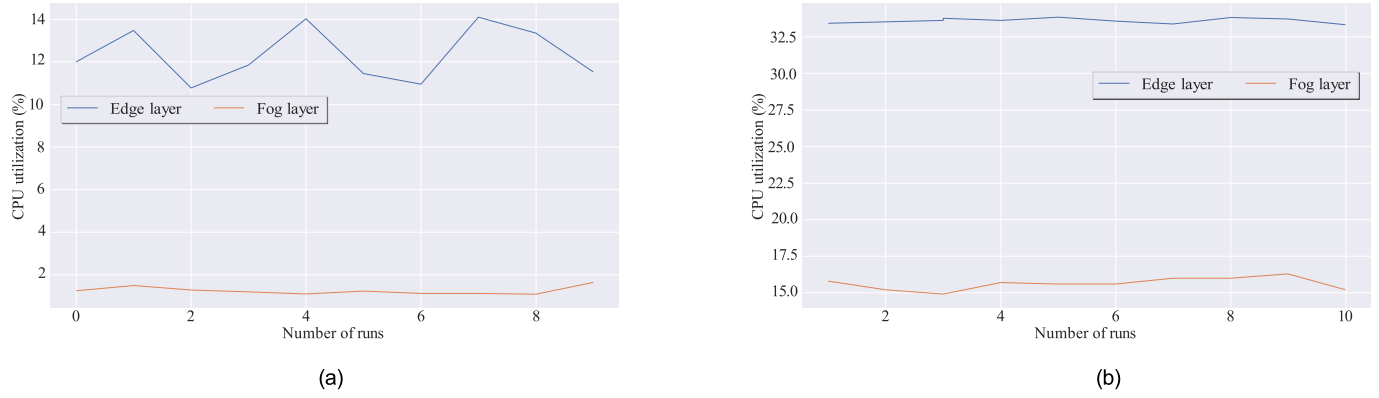

Fig. 13. CPU utilization. (a) Average CPU utilization to perform data transformation and migration operations at the edge and fog layers. (b) Average CPU utilization to perform analytics at the edge and fog layers.

Similar to the execution time, we can observe from Figs. 13(a) and 13(b) that the average CPU utilization for the analytics operation is higher than that of the data migration and transformation operations. The average CPU utilization for data migration and transformation operations was around 12.35% at the edge and approximately 1.24% in the fog, which can be observed from Fig. 13(a). On the other hand, the average CPU utilization for analytics on the edge was approximately 33.61%, while it was around 15.62% in the fog, as observed in Fig. 13(b). On a closer inspection of the CPU utilization for the data transfer and migration operations, we observe low CPU utilization, indicating that data transfer and migration operations are not resource-intensive compared to analytics operation. Moreover, the CPU utilization in the case of the analytics operation is almost constant. However, in the case of data transformation and migration, particularly for the edge layer, we see ups and downs in execution time. These spikes are due to the data transformation operation performing the transformation of the sensor data into the KG representation. In general, the overall CPU utilization for the analytics operation is less than 40% and therefore can be considered normal utilization. Similar to the study of memory usage, the evaluation of CPU usage illustrates the feasibility of the proposed framework.

Fig. 14(a), on the other hand, shows the latency in getting the analytics result at the edge after performing analytics operations in the fog. On average, it took around 1.45 s to get the analytics result at the edge of the fog. The certain high spikes in Fig. 14(a) indicate a higher delay, which could have been caused by network issues such as congestion. This time is significantly less than the time required to perform at the edge itself, highlighting that it is not always good to perform all operations at the edge. Similarly, Fig. 14(b) shows the time taken to create the blockchain transactions, which on average is 0.22 s. Similar to the spikes in Fig. 14(a), we can see that certain transaction times are higher than average (green line), which is highlighted in red.

In summary, from our evaluation, we draw the following major conclusion.

1. In comparison to data transmission and migration procedures at the edge and in the fog, the analytics process is time-consuming. Therefore, performance optimization should concentrate on analytics rather than transfer and migration activities, especially at the edge.
2. Memory use is minimal for both analytic and data transfer and migration processes. Likewise, the CPU usage for the data transfer and migration operations is low, while the CPU usage for the analytics operation is normal. This low resource consumption shows that the proposed framework can be deployed on devices with limited resources.
3. The latency measurement shown in Fig. 14(a) indicates that it is not always advisable to perform all operations at the edge; rather, it is advantageous to delegate compute-intensive tasks to the closest fog. In addition, Fig. 14(a) also highlights the importance of the edge, fog, and cloud hierarchy.
4. Finally, the evaluation validates our claim that the proposed framework is suitable for the edge and fog situation.

7.2.3. *Data quality and veracity evaluation*

We also conducted an evaluation of the data quality, data veracity (or accuracy) and also an assessment of the context information as part of the data quality. The data quality and also the context information assessment are made using SHACL, and the data veracity (or accuracy) is determined by comparing the generated data hash, as discussed

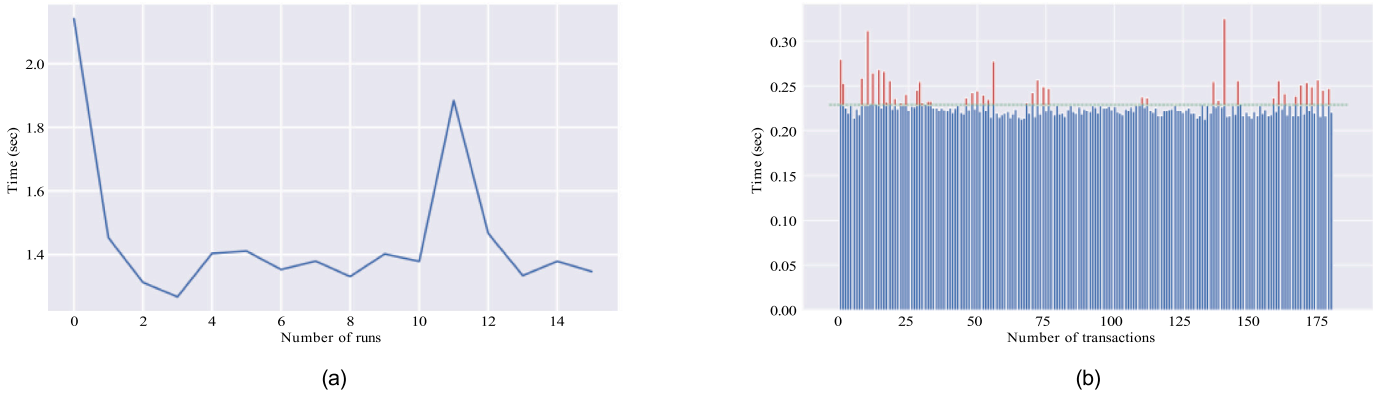

Fig. 14. Execution time. (a) Time taken to send the data to fog and get the analytics result at edge. (b) Time taken to create the blockchain transaction.

in the implementation section (see Section 6.2). The evaluation and assessment of the data quality were conducted manually by checking the generated sensor observations and the shared data after performing the data quality check using SHACL. In particular, the following criteria were checked: (i) the presence of context information such as observation units and the property being observed (i.e., the *sosa* : *observedProperty* value); (ii) the presence of an observation value with a defined data type; and (iii) the presence of the blockchain hash value and trust score with a defined data type. There were a total of 180 observations examined from our edge and edge/fog experiment, and all of the shared data met the defined quality criteria, including the evaluation of context information, resulting in 100 percent accuracy. A similar assessment was made in the case of the data veracity check, which involved comparing the hash of the original data. With this evaluation result, we can confirm that our proposed framework allows for the sharing of high-quality data and the verification of its veracity.

7.2.4. Interoperability evaluation

The interoperability evaluation is conducted qualitatively following the study by Koo & Kim [50] and Cimmino et al. [51]. The interoperability was evaluated using the following criteria: if the proposed system translates heterogeneous raw sensor data into a semantically interoperable format using KG representations in RDF serialization expressed according to ontology, derived based on [51]. This is because ontology formalizes the definition of concepts and represents their shared meaning. Moreover, an additional manual validation is also performed, similar to Cimmino et al.'s [51] work. The evaluation concludes with a verdict that the proposed system supports interoperability following 180 observations (see Section 7.2.3) as the system can translate the heterogeneous data into KG representation based on ontology. In the case of context-dependent variability, interoperability will be restricted, and addressing the issue of context-dependent variability of concepts requires reusing the existing ontology rather than developing a new one and adhering to ontology engineering best practices, such as providing definitions for ambiguous concepts.

7.2.5. Evaluation of analytics operations

Aside from the performance evaluation (see Section 7.2.2), we also evaluated the analytics result by inspecting the SWRL reasoning outcome to determine whether the correct result was attained. A total of 40 observations in the edge case and 14 in the fog–edge scenario were inspected. In the case of the edge, it was anticipated that all 40 observations would activate the alert, i.e., qualify as a temperature or humidity alert. In the case of the fog–edge scenario, seven observations were within the threshold; therefore, only seven of the fourteen should have triggered an alert. According to the results of our evaluation, SWRL reasoning is executed accurately in every scenario.

8. Conclusion

In this research, we investigate IoT data sharing problems and proposed a novel framework that can IoT data sharing challenges of interoperability, accuracy, and quality IoT data sharing in a privacy-conscious manner i.e., adhering to legal restrictions such as GDPR. We demonstrated how ontologies and KGs can address interoperability (see Section 5.4) and support intelligence (see Section 5.7) in the edge/fog scenario. Moreover, we demonstrated how we can ensure the quality (see Section 5.6) of the data being shared through the use of SHACL, enable trustworthiness (see Section 4.2) through the proposed intrinsic trust metric, and achieve accuracy (or veracity) through the use of blockchain in a GDPR-compliant manner.

In addition, we validated our claims, such as suitability for the fog and edge, by implementing the proposed framework, conducting experiments with two distinct scenarios (see Section 7.1), and evaluating their practicality. Our evaluation (see Sections 7.2.2–7.2.5) supports the applicability of the proposed framework (or framework), hence providing support for our claims and answering our research questions (see Section 2.1). Moreover, the comparison of our work to state-of-the-art works (see Table 1) demonstrates additional benefits that our work provides.

The true advantage of the proposed framework lies in its extensibility to cover larger heterogeneous domains at scale, for example, by extending the scope of the ontology used. The proposed framework also ensures the accuracy and quality of the data (see Sections 7.2.3 and 6.2.5), which is an additional benefit and is essential when making data-driven decisions and capitalizing on IoT data and enables privacy-aware data sharing and processing following data protection regulations such as GDPR–one of the limiting factor in IoT data sharing and processing. This is another benefit of the proposed framework as the framework addresses the hindrance posed by laws like GDPR. In addition, the introduction of the trust metric allows one to determine how much the data can be trusted, the other benefit of our work. Using the proposed solution, future work would consist of investigating interoperability and integration at scale in real-world deployments.

CRediT authorship contribution statement

Tek Raj Chhetri: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Chinmaya Kumar Dehury:** Investigation, Writing – review & editing. **Blesson Varghese:** Investigation, Writing – review & editing. **Anna Fensel:** Funding acquisition, Investigation, Writing – review & editing. **Satish Narayana Srirama:** Investigation, Writing – review & editing. **Rance J. DeLong:** Investigation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All the resources, code, ontology are made available.

Acknowledgments

This research has been supported by the European Union projects funded under Horizon 2020 research and innovation program (smashHit), grant agreement 871477 and in part by the European Union Horizon Europe project UPCAST under grant agreement number 101093 216. We would like to express our appreciation to Simon Außerlechner, system engineer at STI Innsbruck, for facilitating temporary servers for experimentation. Additionally, we would like to acknowledge the anonymous reviewer whose comment has helped to improve the manuscript significantly.

Appendix A. Supplementary data

The preliminaries of the concepts, such as KGs, used in this paper are available in the supplementary material.

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.future.2024.03.039.

References

[1] R. Woodhead, P. Stephenson, D. Morrey, Digital construction: From point solutions to IoT ecosystem, Autom. Constr. 93 (2018) 35–46.
[2] Y. Hajjaji, W. Boulila, I.R. Farah, I. Romdhani, A. Hussain, Big data and IoT-based applications in smart environments: A systematic review, Comp. Sci. Rev. 39 (2021) 100318.
[3] M. Centenaro, C.E. Costa, F. Granelli, C. Sacchi, L. Vangelista, A survey on technologies, standards and open challenges in satellite IoT, IEEE Commun. Surv. Tutor. 23 (3) (2021) 1693–1720.
[4] N. Shabana, G. Velmathi, Advanced tele-surgery with IoT approach, in: D. Thalmann, N. Subhashini, K. Mohanaprasad, M.S.B. Murugan (Eds.), Intelligent Embedded Systems, Springer Singapore, Singapore, 2018, pp. 17–24.
[5] J. Byabazaire, G. O'Hare, D. Delaney, Data quality and trust: Review of challenges and opportunities for data sharing in IoT, Electronics 9 (12) (2020) 2083.
[6] M. Bansal, I. Chana, S. Clarke, A survey on IoT big data: Current status, 13 V's challenges, and future directions, ACM Comput. Surv. 53 (6) (2020).
[7] A.A. Alwan, M.A. Ciupala, A.J. Brimicombe, S.A. Ghorashi, A. Baravalle, P. Falcarin, Data quality challenges in large-scale cyber-physical systems: A systematic review, Inf. Syst. 105 (2022) 101951.
[8] L. Ehrlinger, W. Wöß, A survey of data quality measurement and monitoring tools, Front. Big Data 5 (2022) 850611.
[9] M. Naeem, T. Jamal, J. Diaz-Martinez, S.A. Butt, N. Montesano, M.I. Tariq, E. De-la Hoz-Franco, E. De-La-Hoz-Valdiris, Trends and future perspective challenges in big data, in: J.-S. Pan, V.E. Balas, C.-M. Chen (Eds.), Advances in Intelligent Data Analysis and Applications, Springer Singapore, Singapore, 2022, pp. 309–325.
[10] A.M. Ouksel, A. Sheth, Semantic interoperability in global information systems, SIGMOD Rec. 28 (1) (1999) 5–12.
[11] J. Manyika, M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin, D. Aharon, The Internet of Things: Mapping the Value Beyond the Hype, McKinsey Global Institute, 2015.
[12] A. Sharma, E.S. Pilli, A.P. Mazumdar, P. Gera, Towards trustworthy internet of things: A survey on trust management applications and schemes, Comput. Commun. 160 (2020) 475–493.
[13] G. Fortino, L. Fotia, F. Messina, D. Rosaci, G.M.L. Sarné, Trust and reputation in the internet of things: State-of-the-art and research challenges, IEEE Access 8 (2020) 60117–60125.
[14] European Parliament and Council, Regulation (EU) 2016/679 of the European parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/EC (general data protection regulation), Off. J. Eur. Union L119 (May 2016).
[15] M. Bansal, M. Nanda, M.N. Husain, Security and privacy aspects for internet of things (IoT), in: 6th International Conference on Inventive Computation Technologies, ICICT, 2021, pp. 199–204.
[16] J. Byabazaire, G. O'Hare, D. Delaney, Using trust as a measure to derive data quality in data shared IoT deployments, in: 29th International Conference on Computer Communications and Networks, ICCCN, 2020, pp. 1–9.
[17] R. Guha, Search result ranking based on trust, 2009, US Patent 7, 603, 350.
[18] S. Mishra, S. Jain, Ontologies as a semantic model in IoT, Int. J. Comput. Appl. 42 (3) (2020) 233–243.
[19] D. Puthal, S.P. Mohanty, E. Kougianos, G. Das, When do we need the blockchain? IEEE Consum. Electron. Mag. 10 (2) (2021) 53–56.
[20] L. Yang, W. Zou, J. Wang, Z. Tang, EdgeShare: A blockchain-based edge data-sharing framework for industrial internet of things, Neurocomputing 485 (2022) 219–232.
[21] C.K. Dehury, S.N. Srirama, T.R. Chhetri, CCoDaMiC: A framework for coherent coordination of data migration and computation platforms, Future Gener. Comput. Syst. 109 (2020) 1–16.
[22] T.R. Chhetri, A. Kurteva, R.J. DeLong, R. Hilscher, K. Korte, A. Fensel, Data protection by design tool for automated GDPR compliance verification based on semantically modeled informed consent, Sensors 22 (7) (2022).
[23] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, J.P. Jue, All one needs to know about fog computing and related edge computing paradigms: A complete survey, J. Syst. Archit. 98 (2019) 289–330.
[24] J.N.S. Rubí, P.R. de Lira Gondim, IoT-based platform for environment data sharing in smart cities, Int. J. Commun. Syst. 34 (2) (2021) e4515.
[25] F. Loukil, C. Ghedira-Guegan, A.-N. Benharkat, PATRIoT: A data sharing platform for IoT using a service-oriented approach based on blockchain, in: E. Kafeza, B. Benatallah, F. Martinelli, H. Hacid, A. Bouguettaya, H. Motahari (Eds.), Service-Oriented Computing, Springer International Publishing, Cham, 2020, pp. 121–129.
[26] F. Loukil, C. Ghedira-Guegan, K. Boukadi, A.N. Benharkat, LIoPY: A legal compliant ontology to preserve privacy for the internet of things, in: IEEE 42nd Annual Computer Software and Applications Conference, COMPSAC, Vol. 02, 2018, pp. 701–706.
[27] J. Strassner, W.W. Diab, A semantic interoperability architecture for internet of things data sharing and computing, in: IEEE 3rd World Forum on Internet of Things, WF-IoT, 2016, pp. 609–614.
[28] R. Reda, F. Piccinini, G. Martinelli, A. Carbonaro, Heterogeneous self-tracked health and fitness data integration and sharing according to a linked open data approach, Computing 104 (4) (2022) 835–857.
[29] M. Zappatore, A. Longo, A. Martella, B. Di Martino, A. Esposito, S.A. Gracco, Semantic models for IoT sensing to infer environment–wellness relationships, Future Gener. Comput. Syst. 140 (2023) 1–17.
[30] I. Makhdoom, I. Zhou, M. Abolhasan, J. Lipman, W. Ni, PrivySharing: A blockchain-based framework for privacy-preserving and secure data sharing in smart cities, Comput. Secur. 88 (2020) 101653.
[31] S. Abdullah, J. Arshad, M.M. Khan, M. Alazab, K. Salah, PRISED tangle: a privacy-aware framework for smart healthcare data sharing using IOTA tangle, Complex Intell. Syst. (2022).
[32] M. Divya, N.B. Biradar, IOTA-next generation block chain, Int. J. Eng. Comput. Sci. 7 (2018) 23823–23826.
[33] P. Bai, S. Kumar, K. Kumar, O. Kaiwartya, M. Mahmud, J. Lloret, GDPR compliant data storage and sharing in smart healthcare system: A blockchain-based solution, Electronics 11 (20) (2022) 3311.
[34] B. Alamri, I.T. Javed, T. Margaria, A GDPR-compliant framework for IoT-based personal health records using blockchain, in: 11th IFIP International Conference on New Technologies, Mobility and Security, NTMS, 2021, pp. 1–5.
[35] K.M. Tsiouris, D. Gatsios, V. Tsakanikas, A.A. Pardalis, I. Kouris, T. Androutsou, M. Tarousi, N.V. Sedlar, I. Somarakis, F. Mostajeran, N. Filipovic, H. op den Akker, D.D. Koutsouris, D.I. Fotiadis, Designing interoperable telehealth platforms: bridging IoT devices with cloud infrastructures, Enterp. Inf. Syst. 14 (8) (2020) 1194–1218.
[36] D.K. Halim, S. Hutagalung, Towards data sharing economy on internet of things: a semantic for telemetry data, J. Big Data 9 (1) (2022) 1.
[37] S.R. Poojara, C.K. Dehury, P. Jakovits, S.N. Srirama, Serverless data pipeline approaches for IoT data in fog and cloud computing, Future Gener. Comput. Syst. 130 (2022) 91–105.
[38] J. McChesney, N. Wang, A. Tanwer, E. de Lara, B. Varghese, Defog: Fog computing benchmarks, in: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, SEC '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 47–58.
[39] C. Liu, S. Guo, S. Guo, Y. Yan, X. Qiu, S. Zhang, LTSM: Lightweight and trusted sharing mechanism of IoT data in smart city, IEEE Internet Things J. 9 (7) (2022) 5080–5093.
[40] F. Qaswar, M. Rahmah, M.A. Raza, A. Noraziah, B. Alkazemi, Z. Fauziah, M.K.A. Hassan, A. Sharaf, Applications of ontology in the internet of things: A systematic analysis, Electronics 12 (1) (2022) 111.
[41] N.F. Noy, D.L. McGuinness, Ontology development 101: a guide to creating your first ontology, 2001.

[42] T.R. Chhetri, C.K. Dehury, B. Varghese, A. Fensel, S.N. Srirama, R.J. DeLong, Code: Enabling privacy-aware interoperable and quality IoT data sharing with context, 2022, https://github.com/tekrajchhetri/secure-interoperable-data-sharing. (Last accessed 01.12.2022).
[43] K. Janowicz, A. Haller, S.J. Cox, D. Le Phuoc, M. Lefrançois, SOSA: A lightweight ontology for sensors, observations, samples, and actuators, J. Web Semant. 56 (2019) 1–10.
[44] H. Rijgersberg, M. van Assem, J. Top, Ontology of units of measure and related concepts, Semant. Web 4 (2013) 3–13, 1.
[45] D. Bonino, F. Corno, DogOnt - ontology modeling for intelligent domotic environments, in: A. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. Finin, K. Thirunarayan (Eds.), The Semantic Web, ISWC 2008, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 790–803.
[46] S.E.A. Rafey, A. Abdel-Hamid, M.A. El-Nasr, CBSTM-IoT: Context-based social trust model for the internet of things, in: International Conference on Selected Topics in Mobile & Wireless Networking, MoWNeT, 2016, pp. 1–8.
[47] A. AlHogail, Improving IoT technology adoption through improving consumer trust, Technologies 6 (3) (2018) 64.
[48] S. Gupta, Non-functional requirements elicitation for edge computing, Internet Things 18 (2022) 100503.
[49] V. Cheval, C. Cremers, A. Dax, L. Hirschi, C. Jacomme, S. Kremer, Hash gone bad: Automated discovery of protocol attacks that exploit hash function weaknesses, in: 32nd USENIX Security Symposium, USENIX Security 23, USENIX Association, Anaheim, CA, 2023, pp. 5899–5916.
[50] J. Koo, Y.-G. Kim, Resource identifier interoperability among heterogeneous IoT platforms, J. King Saud Univ. - Comput. Inf. Sci. 34 (7) (2022) 4191–4208.
[51] A. Cimmino, J. Cano-Benito, A. Fernández-Izquierdo, C. Patsonakis, A.C. Tsolakis, R. García-Castro, D. Ioannidis, D. Tzovaras, A scalable, secure, and semantically interoperable client for cloud-enabled demand response, Future Gener. Comput. Syst. 141 (2023) 54–66.
[52] T.R. Chhetri, A. Kurteva, J.G. Adigun, A. Fensel, Knowledge graph based hard drive failure prediction, Sensors 22 (3) (2022) 985.

Tek Raj Chhetri is a Postdoctoral Associate at the Senseable Intelligence Group at the McGovern Institute for Brain Research at the Massachusetts Institute of Technology and the founder and director of CAIR-Nepal (Center for Artificial Intelligence (AI) Research Nepal), an artificial intelligence research organization in Nepal. He holds a Ph.D. degree in Computer Science from the University of Innsbruck, Austria, and a Master's degree in Computer Science from the University of Tartu, Estonia. He was awarded a full-time exemption scholarship from Tartu University, a grant from the Estonian Ministry of Foreign Affairs, and a Kristjan Jaak scholarship from the Archimedes Foundation in Tartu, Estonia. He worked as a research assistant on an autonomous vehicle at the Intelligent Transportation System Lab, Tartu, and as a software developer at Auve Tech. His research interests include Knowledge Graphs, Artificial Intelligence, Privacy, the Internet of Things, Edge Intelligence, and Distributed Systems. More information is available from https://tekrajchhetri.com or https://sites.mit.edu/tekrajchhetri.

Chinmaya Kumar Dehury received bachelor degree from Sambalpur University, India, in June 2009 and MCA degree from Biju Pattnaik University of Technology, India, in June 2013. He received the Ph.D. Degree in the department of Computer Science and Information Engineering, Chang Gung University, Taiwan. Currently, he is a postdoctoral research fellow in the Mobile & Cloud Lab, Institute of Computer Science, University of Tartu, Estonia. His research interests include scheduling, resource management and fault tolerance problems of Cloud and fog Computing, and the application of artificial intelligence in cloud management. He is an reviewer to several journals and conferences, such as IEEE TPDS, IEEE JSAC, Wiley Software: Practice and Experience, etc.

Blesson Varghese received the Ph.D. degree in computer science from the University of Reading, UK on international scholarships. He is a Reader in computer science at the University of St Andrews, UK, and the Principal Investigator of the Edge Computing Hub. He is an Honorary faculty member at Queen's University Belfast and a previous Royal Society Short Industry Fellow. His interests include distributed systems that span the cloud–edge-device continuum and edge intelligence applications. More information is available from www.blessonv.com.

Anna Fensel has a university degree in mathematics and computer science from Novosibirsk State University, Russia. She received her doctoral degree and her habilitation in computer science at the University of Innsbruck, Austria. She worked as a Research Fellow at the University of Surrey, UK, Senior Researcher at FTW — Telecommunications Research Centre Vienna, and as Assistant and then Associate Professor at the University of Innsbruck, Austria. Since 2022, she is a Professor at Wageningen University and Research, the Netherlands. She has been on various committees of more than 100 scientific events, and authored more than 120 refereed publications.

Satish Narayana Srirama is a Professor at the School of Computer and Information Sciences, University of Hyderabad, India. He is also a Visiting Professor and the honorary head of the Mobile & Cloud Lab at the Institute of Computer Science, University of Tartu, Estonia, which he led as a Research Professor until June 2020. He received his Ph.D. in computer science from RWTH Aachen University, Germany in 2008. His current research focuses on cloud computing, mobile web services, mobile cloud, Internet of Things, fog computing, migrating scientific computing and enterprise applications to the cloud and large-scale data analytics on the cloud. He is IEEE Senior Member, an Editor of Wiley Software: Practice and Experience, a 52 year old Journal, was an Associate Editor of IEEE Transactions in Cloud Computing and a program committee member of several international conferences and workshops. Dr. Srirama has co-authored over 175 refereed scientific publications in international conferences and journals. For further information of Prof. Srirama, please visit: https://scis.uohyd.ac.in/~srirama/.

Rance J. DeLong is Staff Scientist at The Open Group, with 40+ years of software experience, having particular interest in methods and tools for compositional construction and assurance of critical systems. He has contributed to the design of numerous secure operating systems and a MILS-based trusted smart phone. Rance has participated in 10 EC research projects, including D-MILS, CITADEL, Cross-CPP and smashHit addressing assurance, security and privacy for cyber–physical systems, and he has been a contributor to MILS research for 18 years. Rance holds three patents, degrees in Physics and Philosophy, with extensive postgraduate study in Computer Science.