applied sciences

Article

Privacy-Aware and Secure Decentralized Air Quality Monitoring

Michael Mrissa [1,2,*,†], Aleksandar Tošić [1,2,†], Niki Hrovatin [1,2,†], Sidra Aslam [1,2,†], Balázs Dávid [1,2,†], László Hajdu [1,2,†], Miklós Krész [1,3,†], Andrej Brodnik [2,4,†] and Branko Kavšek [2,5,†]

1 InnoRenew CoE, Livade 6, 6310 Izola, Slovenia; aleksandar.tosic@innorenew.eu (A.T.); niki.hrovatin@innorenew.eu (N.H.); sidra.aslam@innorenew.eu (S.A.); balazs.david@innorenew.eu (B.D.); laszlo.hajdu@innorenew.eu (L.H.); miklos.kresz@innorenew.eu (M.K.)
2 Faculty of Mathematics, Natural Sciences and Information Technology (FAMNIT), University of Primorska, Glagoljaška 8, 6000 Koper, Slovenia; andrej.brodnik@fri.uni-lj.si (A.B.); branko.kavsek@upr.si (B.K.)
3 Institute Andrej Marušič (IAM), University of Primorska, Muzejski trg 2, 6000 Koper, Slovenia
4 Faculty of Computer and Information Science, University of Ljubljana, Večna pot 113, 1000 Ljubljana, Slovenia
5 Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
* Correspondence: michael.mrissa@innorenew.eu
† These authors contributed equally to this work.

Abstract: Indoor Air Quality monitoring is a major asset to improving quality of life and building management. Today, the evolution of embedded technologies allows the implementation of such monitoring on the edge of the network. However, several concerns need to be addressed related to data security and privacy, routing and sink placement optimization, protection from external monitoring, and distributed data mining. In this paper, we describe an integrated framework that features distributed storage, blockchain-based Role-based Access Control, onion routing, routing and sink placement optimization, and distributed data mining to answer these concerns. We describe the organization of our contribution and show its relevance with simulations and experiments over a set of use cases.

Keywords: air quality monitoring; privacy; security; edge; decentralization

check for updates

Citation: Mrissa, M.; Tošić, A.; Hrovatin, N.; Aslam, S.; Dávid, B.; Hajdu, L.; Krész, M.; Brodnik, A.; Kavšek, B. Privacy-Aware and Secure Decentralized Air Quality Monitoring. *Appl. Sci.* **2022**, *12*, 2147. https://doi.org/10.3390/app12042147

Academic Editor: Prashant Kumar

Received: 20 December 2021
Accepted: 8 February 2022
Published: 18 February 2022

1. Introduction

As we spend most of our time inside buildings, the monitoring of Indoor Air Quality (IAQ) is essential to ensure healthy living [1]. It also greatly contributes to improving building operation and reducing energy consumption [2]. Nowadays, IAQ monitoring systems typically rely on Wireless Sensor Networks (WSNs) that consist of a large number of sensing nodes deployed for specific monitoring purposes. Generally, WSNs transfer the collected data to the cloud for storage and further processing. Such a centralized and cloud-dependent infrastructure represents a security and reliability risk as the connection to the cloud becomes a single point of failure that can be subject to diverse attacks. The risks related to data security and privacy also increase as the storage is remote. With the development of embedded technologies over the last few years, decentralized IAQ monitoring solutions have become appealing as they enable on-site data storage, processing, and analysis [3].

However, decentralized monitoring solutions still suffer from several limitations. First, the collected data require basic security protection as they describe physical phenomena that can reveal the activities happening in a monitored space, for instance, the variation of CO_2 (carbon dioxide) and temperature can indicate human presence in a room. Second, a decentralized storage solution must make sure that data are only accessible to the right stakeholder with sufficient permissions, making privacy a major concern as diverse stakeholders may require access to different views on data. Third, the operation of the developed solution must be protected from external network monitoring that can reveal the activity of involved stakeholders, the purpose of the WSN, and the contents of the exchanged messages. Fourth, the complexity of such a solution must be reduced as much as possible in order to efficiently handle larger amounts of data, number of sensors, and amount of communication. Although these problems belong to the same scenario, they are generally addressed individually. Therefore, we provide in this paper a decentralized framework for IAQ monitoring that integrates different components to address the aforementioned limitations. We present our contributions and show how they work together by describing their decentralized implementations on sensors.

Section 2 describes the use cases that motivate our work and shows the relevance of our contribution. Section 3 overviews the most relevant related work and highlights the originality of our approach, with respect to the general framework and each individual contribution. Section 4 details our decentralized framework together with the role and operation of each component. Section 5 discusses the results of our experiments. Finally, Section 6 summarizes our obtained results and gives guidelines for future work.

2. Motivating Scenario

Our motivating scenario is based on a set of five selected buildings that form a starting point towards a federation of smart buildings that contribute to building joint knowledge about IAQ. Each building contributes to optimizing its own life cycle and to extending common knowledge by sharing its monitoring model with other buildings. The pilot buildings include:

- The Danilo Lokar primary school, located in Ajdovščina, Slovenia;
- The Peavey hall research laboratory, located in Corvallis, Oregon, USA;
- The Nobatek research building, located in Anglet, France;
- The InnoRenew research building, located in Izola, Slovenia;
- The AlbaComp company offices, located in Szeged, Hungary.

The data coming from these buildings require privacy, security, and independence from the cloud. As a public institution, the primary school requires an IAQ monitoring that does not reveal information about pupils and teachers. The three research laboratories need to keep their monitoring activities as well as building occupation patterns away from public view. The private research company needs independence from the cloud to assess its operation without any trusted third party.

However, all five pilots would benefit from enabling and exploiting common knowledge that would enable better building operation and management and improved efficiency in classrooms and offices. The main element that the multiple models have to overcome is the different climate conditions the buildings are submitted to due to their geographical location. Therefore, sharing models enables the identification of which factors responsible for building behavior are local and which are not, which helps in identifying which measures are only applicable to local buildings or to all.

This scenario motivates the need for autonomous communities of sensors that, locally, collaborate to improve building operation and well being and, globally, contribute to improving the prediction of building conditions. In the following, we review the most relevant related works that connect to our contributions.

3. Related Work

In this section, we first overview previous works that compare to ours with the same objective—IAQ monitoring—and identify the research gaps that justify the need for our work. Then, we overview existing work related to the different parts of our contribution, namely, decentralized data mining, decentralized privacy-aware data storage, secure data processing, and, finally, route optimization and sink placement.

3.1. Decentralized IAQ Monitoring Frameworks

A considerable amount of work is dedicated to building IoT systems for IAQ monitoring, as shown in a systematic review published quite recently [4]. Some solutions rely on lightweight communication protocols such as LORAWAN technology [5] or ZigBee [6] to alleviate the processing load from the sensor and enable the use of small, less resource-consuming devices. However, most solutions send data to the cloud for further processing due to the constrained nature of those IoT devices. Although the work in [4] addresses several aspects of IAQ monitoring, the parts that lie in the scope of this paper, related to data storage, show that most solutions (65%) rely on cloud storage to store the data, the others relying on non-decentralized IoT data storage solutions.

In [7], the authors propose a decentralized IAQ solution based on IOTA and Masked Authenticated Messaging (MAM). One limitation of this work is the relative centralization problems that come with IOTA. Another significant difference with our work is the lack of privacy management, task protection from monitoring, and optimization.

In [8], the authors present a decentralized IoT data storage framework using IOTA Tangle and IPFS (InterPlanetary File System). The proposed framework is based on a central data handling unit (such as a local server) to collect and encrypt the data, which is subject to a single point of failure. The encrypted data are sent to the IPFS, while the corresponding hash and metadata are stored on the IOTA Tangle. However, in addition to the aforementioned limitations of IOTA, the IPFS network is immutable, which does not allow users to change the data after storing them [9].

The authors in [10] presented a framework that integrated the blockchain with IoT-enabled air quality monitoring system. The proposed framework used the blockchain to store the air pollution data of industries in the smart city. However, the security aspects are not investigated.

In [11], the authors provide an Azure cloud-based system to predict air quality and give a warning when the air quality status needs to change. The proposed system collects air quality data and then predicts the quality of air pollution. The air quality data are stored in the PostgreSQL database (built from the Azure cloud platform) and at the same time replicates data on the blockchain nodes to ensure data transparency. Such a combination raises several limitations, such as a lack of scalability of the blockchain for large quantities of data, lack of data privacy and fine-grained access control, and single point of failure for the PostgreSQL database.

As far as we could see from the existing literature, a fully decentralized IAQ monitoring solution that integrates distributed data storage, fine-grained access control, integrated data mining, secure task execution, and route optimization is still lacking, thus motivating the contribution described in Section 4.

3.2. Decentralized Data Mining

While conventional data mining (DM) approaches rely on data available at a single location, the key challenge of decentralized data mining (DDM) is to use “decentralized coordination with local decision making to achieve the intended global goal” [12].

In [13], a survey of state-of-the-art DDM techniques is provided, including distributed frequent itemset mining, distributed frequent sequence mining, distributed frequent graph mining, distributed clustering, and privacy-preserving distributed data mining (PPDDM).

While [13] provides a relatively recent survey of DDM techniques with emphasis also on privacy preservation, Ref. [14] introduces the notion of decentralized spatial data mining (DSDM), where individual sensor-enabled computing nodes possess only local knowledge about their immediate neighborhood but derive global knowledge through local collaboration and information exchange. The latter is especially relevant for our research, where we aim at monitoring IAQ over a WSN.

3.3. Security of WSN Data Processing

The task of processing the data sourcing from a network of sensors is usually carried out in an external system, often by relying on cloud services. Therefore, the sensor node data must be transferred to the external system; commonly, the data transmission is protected using Transport Layer Security (TLS). However, the data in a WSN travel through a multi-hop infrastructure-less network, and using TLS could rapidly overload the resource-constrained technology WSNs consist of. Moreover, the processing power of nodes is not utilized, and several studies point out the possibility of discovering associations between TLS traffic patterns and activities in the monitored environment [15–17].

Additionally, gathering data from all sensor nodes in the WSN may be redundant since closely located sensor nodes might sense the identical event or changes. Data redundancy can be reduced by aggregating readings of closely located sensor nodes. The in-network data aggregation technique works via the aggregation of sensor node's readings as the data travel through routing paths toward the sink node. Therefore, the sink node receives an aggregated value summarizing the state of the monitored area. Several [18] privacy-preserving data aggregation solutions were developed relying on data perturbation, data slicing, hop-by-hop encryption, and Homomorphic Encryption [19].

Data perturbation techniques [20] modify individual sensor reading values to hide specific confidential information but still allowing users to retrieve the correct aggregated value. In [21], twin-keys are used to add or subtract a shadow value to the sensor reading. The aggregated value is correctly computed since any shadow value added by one node during the aggregation process is subtracted by another node that shares the same twin-key.

The data slicing consists of slicing single sensor node data into a fixed number of data pieces, all data pieces except one are sent to neighboring nodes. Nodes assemble data pieces including their own piece and send the assembled data to an aggregator node [22,23]. However, data slicing leads to high communication overhead due to the large number of data pieces sent to neighboring nodes.

In hop-by-hop data aggregation, sensor node readings are decrypted and encrypted at each node in the aggregation path. Therefore, aggregator nodes can access the unencrypted data. Several solutions [24,25] apply a data perturbation technique to protect the unencrypted data from malicious aggregator nodes.

The Homomorphic Encryption was used in several solutions [26–28] since it allows operations to be performed on encrypted data without the need for decryption. However, the Homomorphic Encryption is computationally intensive [29,30], and reasonable solutions for WSNs are designed to only compute addition and multiplication on encrypted data.

Secure Multi-Party Computation (SMC) [31] allows multiple parties to jointly compute a function over their inputs while keeping those inputs private. Several studies propose the SMC technique for privacy preserving data mining since using the SMC for training a data mining model preserves the data privacy of individual data sources. Refs [32,33] describe an SMC technique for privacy-preserving association rule mining; in [34], an SMC decision-tree-based data classification was proposed; and the authors in [35] proposed a protocol for securely computing a linear regression model. However, traditional solutions relying on SMC require high computation and communication costs, thus being not convenient for application in WSNs. Therefore, the SMC in WSNs is limited to low complexity tasks such as data aggregation [36].

The presented decentralized framework provides data processing security using the General Purpose Data and Query Privacy Preserving Protocol for WSN presented in [37]. The technique uses messages composed of several encryption layers to convey on WSN nodes general-purpose computer code for in situ data processing. Data processing results from multiple nodes along the message path are aggregated, and the privacy of data is protected by encryption and by concealing the identity of nodes performing data processing among a set of nodes that bogus data processing.

3.4. Decentralized Privacy-Aware Data Storage

In this section, we overview the most relevant work that focuses on privacy-aware and decentralized data storage. Most existing solutions rely on a combination of decentralized ledger technology to enable decentralized trust, combined with a Distributed Hash Table (DHT) for data storage.

The authors in [38] propose an Ethereum-based blockchain framework. Although the solution is decentralized, the data storage part relies on a MySQL database to store the data while a hash sum of the data is sent to the blockchain. The main limitation of this work is the MySQL database that becomes a single point of failure. In addition, the hash stored on the Ethereum blockchain leaves information about the data storage operations publicly available.

In [39], the authors propose a blockchain-based framework that allows users to control and keep the track of the data they share with their friends, family, and others. The proposed framework uses a software client to manage the record of the private key that is shared with the circle members. It also encrypts the data with the circle's public key before sharing it. However, the paper mentions private key sharing, which is a major security risk that could lead to key leakage and data security issues.

The authors in [40] develop a food traceability framework based on the blockchain and smart contracts. The proposed framework stores original data on the IPFS, whereas the corresponding hash of the data is stored on the blockchain. A smart contract is used to handle traceability information. However, the proposed framework modules are managed by a manufacturer node server that is subject to a single point of failure. In addition, data stored on the smart contract are immutable and cannot be updated later. One more limitation of the IPFS network is the immutability that stores file permanently [9]. On the other hand, we use DHT that allows modification in the data after storing it in the database.

In [41], the authors combine a blockchain with a Distributed Hash Table (DHT) to manage personal data. A DHT is used to store encrypted data and a symmetric key (a single shared key for encryption/decryption) while the data hash is stored to the blockchain. The proposed framework allows both the service and the user to query data using a data hash. However, the authors did not discuss any solution to protect symmetric keys from unauthorized access. Additionally, a fine-grained access control model is used to access the blockchain. However, the proposed solution records permissions on the blockchain that is subject to immutability concerns.

The authors in [42] propose a blockchain-based data management and access control framework. A blockchain is used to store access control permissions. However, the permissions are stored on the blockchain are immutable and publicly accessible, which raises privacy issues.

A decentralized data storage for PingER (Ping End-to-End Reporting) is presented in [43]. The metadata of PingER files is stored on a permissioned blockchain, while actual data references are stored in a DHT. The work is quite similar to ours but does not consider security aspects.

In summary, the most relevant data storage frameworks rely somehow on a single point of failure at one stage or another of the proposed solution. Another concern is scalability for the solution that stores data on the blockchain. In addition, a good part of them stores data publicly without any security mechanisms, which highlights the need for a privacy-aware, decentralized solution.

3.5. Routing and Sink Placement Optimization

The problem of optimal sink placement requires data about distance from any node to any potential sink. From the perspective of an individual node, the problem is a well-known single-source shortest path problem (SSSP), which has at least two traditional solutions, the Dijkstra's and the Bellman–Ford's solutions ([44]). Both solutions are centralized, which means that the complete network description has to be made available at a node, which performs computation. However, Bellman–Ford's solution can be changed to also work in a distributed environment, where each node is aware only of its direct neighborhood and information in it. This change is built into the Routing Information Protocol (RIP, [45–47]).

In our scenario, initially all nodes are the same and only later are some of them defined as sinks or gateways. In other words, any node can also potentially be a sink. Consequently, we are not solving n, where n is a number of nodes, unrelated SSSP problems, but one single all-pairs shortest path problem (APSP). The traditional algorithm solving this problem is a dynamic-programming centralized Floyd–Warshall algorithm ([44,48]). Indeed, some attempts were made to make it parallel, and most of them applied techniques used in parallel matrix multiplication (cf. [49]).

The network design of a Wireless Sensor Network (WSN) can play a critical role in the effectiveness of the proposed system. The two basic important performance indicators are the coverage and the connectivity. The objective of the coverage is to monitor a set of targets with a subset of sensor nodes, while, without connectivity, the sensor nodes cannot send the measured data to the sink nodes. The literature often combines these two questions in the case of centralized systems [50]. However, in the case of decentralized systems, the objectives can be separated on the network level; no specific approach is requested for coverage, and we need to concentrate on connectivity and communication.

For both centralized and decentralized systems, the optimal placement of the sink nodes and the route optimization have a huge effect on the behavior of the system regarding the main design questions, including robustness [51], production cost [52], power consumption [53], etc. In the case of sink placement, the sensor network is given, and the goal is to identify the sink nodes in an optimal way.

The mathematical formulation of the discussed issues leads back to two graph theoretical problems, the dominating set problem [54] and the facility location problem [55]. Because of the connectivity constraints on the level of the sink nodes, for both problems, the connected version is relevant: the connected dominating set and the connected facility location problems. In the case of a dominating set problem, subproblems, for example, distance dominating set [56] and k-hop connected dominating sets [57], are used in the area of Wireless Sensor Networks. In our case, the problem is very similar; however, our objective is to minimize the distances with a given fixed dominating set size. Regarding dominating sets, several different approaches can be found in the literature [58–60].

The original facility location problem is about placing facilities based on geographical demands in an optimal way in order to minimize facility costs and transportation distances; however, it can also be used in sink placement. The difference in our case is that each node can also be a demand. A similar subproblem of the facility location is the rent or buy problem [61]. Regarding the facility location problem, different approaches can be found in the literature [62–64].

4. A Privacy-Aware and Secure Decentralized Framework

Our contribution consists in a decentralized framework, where each node of the framework is a sensor with some computing capacity. Each node implements our architecture structured into several components that deal with each identified research problems. In the following, we give a general overview of a node architecture and explain the combined operation of our solution. Then, we detail the different parts that contribute to this unique result.

4.1. General Framework Overview

In this section, we discuss the architecture that facilitates the interoperability between aforementioned contributions so that the system can benefit from the desired features of individual modules in an additive way. As shown in Figure 1, the global view of our framework consists of communities of nodes that communicate (1) within a community and (2) between communities, always in a decentralized fashion, which provides supports for multiple levels of interaction.

We assume different configuration of WSNs depending on requirements ranging from private homes, large residential buildings, and public buildings. There should be at least one sink node per WSN deployment. However, depending on the requirements of availability, redundancy can be added to increase the fault tolerance of sink nodes. Sink nodes expose a RESTful API, either to the external or internal network depending on the security requirements. The API serves as an entry point for querying the system, in our use case, building data mining models. Data are secured using cryptographic methods described in the corresponding section. Nodes provide data access under the conditions enabled in the RBAC component of the node. It is then possible to send queries the framework (in our case, monthly average). A query goes through the nodes and realizes the actions necessary to produce the expected result, while using onion routing technique to protect from external monitoring. Upon receiving a request, the sink node prepares the onion so that the path starts and ends at the sink. Knowing the public key of the actor, who initiated the request via the API, the sink node can encrypt the computation results and interact with the blockchain(RBAC) and the DHT. The query results are then stored on the DHT, while the access control is used to secure the pointer to the file on the blockchain. The RBAC is used to limit access to the exposed API on sink nodes and protect the results of the computation so that only the actor who submitted the query can decrypt the result from the DHT.

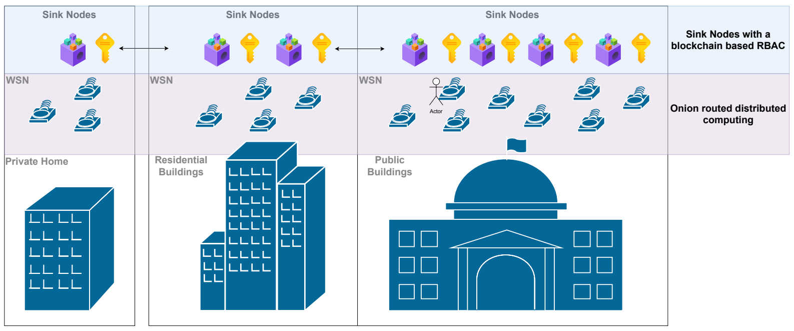

Figure 1. A layered system architecture.

Therefore, the nodes are responsible for preserving privacy and network anonymity between the actors interacting with the API, the underlying WSN, and the computation by onion routing the request as described in Section 4.3. In addition, each node contributes to the decentralized data mining by locally processing the data it collects (for example, precalculating local averages). Each node also participates to optimizing the overall communication by contributing to the network through calculating the best routes as well as the roles of each node.

4.2. Decentralized Data Mining

In this section, we show how data mining can be distributed over a set of nodes. While the majority of existing DDM solutions presented in Section 3.2 rely on computing the machine learning models (ML models) locally at WSN nodes and then combining their knowledge in a global ML model, our proposed approach starts with computing an initial ML model at a (randomly) chosen WSN node and then iteratively "augment" it by updating it with the data at the next WSN node. By repeating this procedure following a (predefined) path through the nodes of the WSN, the last visited node will contain an approximation of the global ML model, as would be computed on all the data from all the WSN nodes on the traversed path.

In this way, instead of resorting to ingenious ways of combining multiple local ML models into one global ML model, we can use techniques from data stream mining (DSM) to iteratively update our initial local ML model at each subsequent WSN node. Moreover, our approach also minimizes the amount of data that has to be sent through the WSN—the descriptions of the local ML models are only shared/sent between the nodes on a predefined path, as opposed to multiple paths that are needed for sending the local ML models to a sink node (for the computation of the global ML model) in the traditional scenario of the "combined" local-to-global ML model computation.

A comprehensive review of the DSM methods is provided by Albert Bifet et al. in their book [65]. We decided to use decision trees as our preferred ML algorithm due to their space/time efficiency and explainability. The Hoeffding tree [66] is a version of the decision tree that can be learned from streams of data. Thus, our DDM method of choice is the Hoeffding tree, which we first learn from the batch data provided by some (randomly) chosen sensor in our WSN and then send through the WSN, following a predefined path of nodes, to be updated by streams of data at each subsequent WSN node. The described procedure is depicted in Figure 2.

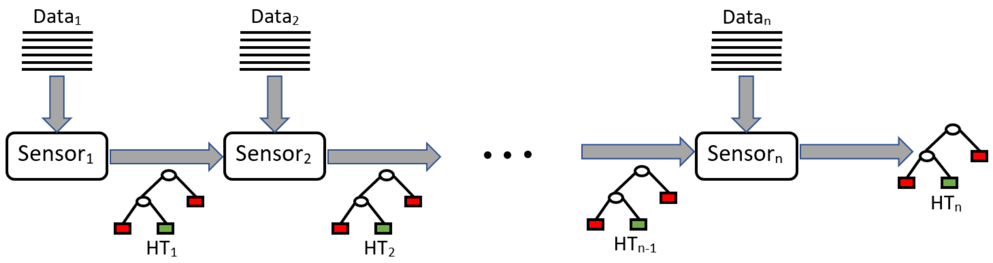

Figure 2. Incremental learning with Hoeffding trees.

The initial Hoeffding tree (HT_1) is first learned in batch from the data provided by the first sensor—$Sensor_1$ ($Data_1$). This tree is then propagated through the WSN, and in each subsequent WSN node ($Sensor_k$), an updated Hoeffding tree is generated (HT_k) from the previous tree (HT_{k-1}) and the data from the current sensor—$Sensor_k$ ($Data_k$). The final decision tree is then represented by the last Hoeffding tree in this chain—HT_n (the output of $Sensor_n$). Notice that only partial Hoeffding trees (HT_k's) are propagated through the WSN (not the actual sensor data), which is most suitable for privacy preserving and also diminishes the load on the WSN.

4.3. Security of WSN Data Processing

The decentralized framework presented in this contribution makes use of the technique described in [37] to distribute the processing load of training a data mining model among sensor network nodes while ensuring the security of data processing. In the following, we will refer to the General Purpose Data and Query Privacy Preserving Protocol for WSNs described in [37] as DQ3P. DQ3P was proposed for retrieving aggregated data from a WSN by securely conveying general-purpose computer code through a set of sensor nodes.

In this contribution, we adopt the DQ3P to deliver data mining models to WSN nodes. Therefore, model training can be carried out in situ without endangering the data privacy of the sensor nodes. DQ3P relies on encryption to deliver the data mining model only to nodes that will carry out model learning. Furthermore, the identity of nodes carrying out model learning is hidden among a set of other nodes that receive the data mining model without encryption keys for accessing it. Thus, the identity of nodes contributing to model learning is known only to the origin, in [37] it is shown that DQ3P is secure even if some nodes associated with the network are intentionally collaborating to disclose data privacy of other sensor nodes. Moreover, DQ3P withstands traffic analysis attacks by generating uniformly distributed network traffic.

The DQ3P defines a particular message; we refer to it as the onion message, which consists of the onion head and the onion body as shown in Figure 3. The onion head is a layered object made of several encryption layers similar to the one used in the Onion Routing [67]. The encryption layers of the onion head are formed using public-key cryptography; therefore, the message must travel through the exact path specified at message construction, each node in the message path unveiling one layer of the layered object. By the DQ3P design, layers of the onion head include: (a) the IP address of the next node in the message path OR (b) the next-hop IP address and two symmetric encryption keys. Onion messages are issued by sink nodes, and message processing at sensor nodes is conditional to the content obtained from layer decryption of the onion head, as follows: (a) Nodes that receive the onion message and obtain only the next-hop IP address from layer decryption of the onion head are not contributing to the model learning. These nodes retain only the onion message for a time interval to simulate model learning task. The onion message is then forwarded to the node at the next-hop IP address. (b) Nodes that also obtain the two symmetric encryption keys from layer decryption of the onion head are nodes that contribute to the model learning. These nodes use the first symmetric encryption key to decrypt the onion body and obtain the data mining model. Model learning is executed on data located on the sensor node, and the updated model is embedded in the onion body. The onion body is then encrypted using the second symmetric encryption key, and the onion message is forwarded to the node at the next-hop IP address.

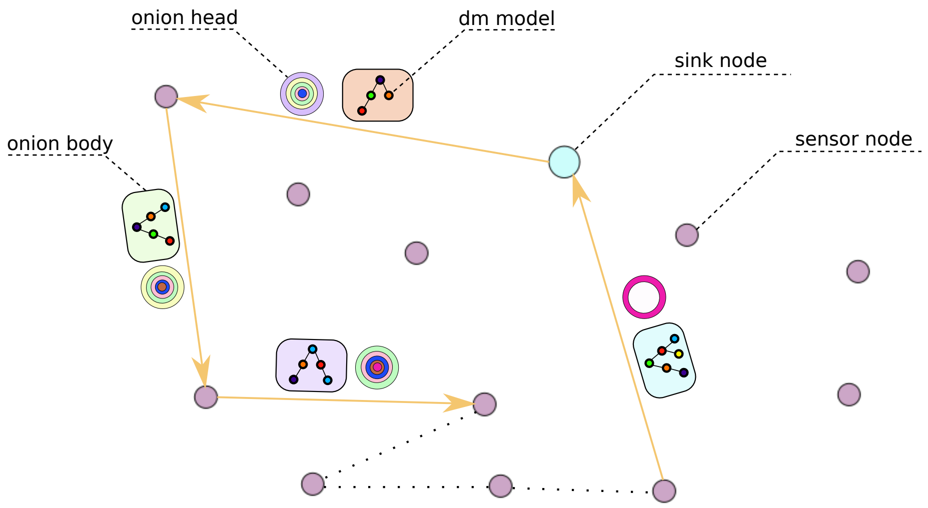

Figure 3. The figure shows the onion message traveling through the network and updating the DM model at sensor nodes in the message path. The onion head and onion body color change at each message processing to indicate the encryption operation.

As shown in Figure 3, onion messages are transiting in the WSN following the path encoded in encryption layers of the onion head and processing model learning only on specific nodes in the message path. The onion message path ends at the same sink node that issued the onion message since the penultimate layer of the onion head always includes the sink's IP address. The last layer of the onion message includes the encryption key to access the content of the onion body and thus obtain the data mining model trained on the data of sensor nodes in the onion message path.

4.4. Decentralized Privacy-Aware Data Storage

As demonstrated in previous work [68], the limitations of blockchain make it necessary to combine it with other technologies to enable fully decentralized storage and at the same time allow for data modifications. In this section, we show a decentralized data storage solution that provides different security levels and privacy protection.

Indeed, data privacy requires addressing the diversity of actors and their roles to provide fine-grained data access and, sometimes, anonymity when disclosing particular data pieces, for example, public statistics about a building. To answer these requirements, we propose a pipeline that combines blockchain with a Distributed Hash Table (DHT), a Role-based Access Control model (RBAC), encryption mechanisms, and a ring signature to manage data access.

Our solution stores metadata and DHT hash keys on the blockchain, while corresponding encrypted data are stored on the DHT that allows the data owners to modify their data.

We propose a metadata structure that includes the date and time of data entry, user ID, DHT key, previous pointer, and data schema URI. We use the RBAC model to verify the actors' permissions to read and write data. In RBAC, a role represents a user's rights to perform actions according to their given permissions. A permission is an authorization to access multi-level data within a domain [69]. RBAC is suitable for our scenario as it allows us to define different roles, such as a building manager, room occupant, or scientist. Each actor has different permission to access the data depending on their role. For instance, with RBAC, we can define that the room occupant can access the fully detailed data values for the room but not others, while for the building manager, only the averages suitable to maintain and operate the building. The scientist could access all values without identification of which room in the building they correspond to.

To complement RBAC, our framework provides different types of encryption mechanisms to secure the data in a decentralized fashion. An authorized actor can choose between asymmetric encryption or symmetric encryption to write and read data. Symmetric encryption is more suitable when a large quantity of data needs to be stored and is expected to be shared with many users, so it is stored encrypted together with the key, and when the data are requested, only the key needs to be decrypted and then encrypted with the requester's public key. Likewise, direct asymmetric encryption is more suitable for small quantities of data that are not expected to be shared with many other users. Please note that in both cases, asymmetric encryption is needed to ensure users' identities, symmetric encryption being an optimization for specific cases.

We use ring signature to hide the actor's identity when sending data. This is useful, for example, when we would like to send data from one sensor within a building without revealing which sensor it is. To sign data, an actor uses their private key and the public keys of other actors. Therefore, no one can identify who signed this data, and it helps to maintain the actor's anonymity within the group. The ring signature is used to ensure the actor's identity privacy. Then, the data requester reads the data and validates the signature to verify that data are coming from an authentic source.

4.5. Routing and Sink Placement Optimization

Our integrated approach concerns the sink placement from the point of view of routing feasibility. Therefore, we first consider the sink placement problem in a formal way. Let $G = (V, E)$ be an undirected connected unweighted graph representing the sensor network. It is important to note that the questions regarding sink placement can be generalized for weighted networks, where the edge weights represent the cost of communication between the nodes. Nevertheless, as the goal is to identify the sink nodes in the network, in the case of a decentralized system, the weights can be also dynamic as the communication cost between the sink nodes can be different from the cost between sensor nodes (or sensor–gateway communication). By the above fact, the generalization in the case of certain conditions is not straightforward.

Our goal is to identify an optimal gateway placement in a graph G with respect to some cost function defined on the network. We assume that the number of k gateways is determined in advance. In different versions of the problem, because of the characteristics of the distributed systems, we need to ensure that all sink nodes (or a certain percentage of the sink nodes) have all the data measured by the sensors. On the theoretical level, we also assume that the system is operating by synchronous steps via data communication executed through edges connecting adjacent nodes. In order to define the sink placement problem, it is important to introduce different methods for the routing paths. We suppose that the routing from each sensor is arranged in a unique way. In practice, multipath routing is possible [70], but it is considered in a stochastic manner (one routing will be ending at a single gateway, but the choices during the routing can be stochastic). In the following, we will distinguish several versions of the problem. In certain cases, multipath routing has no influence on the model, while, in other cases, it could be handled by a stochastic model only with an expected value for each gateway having all the data. This stochastic feature would cause a problem in ensuring the required level of data sharing among the gateways; hence, in this study, we will consider deterministic models only.

First, we suppose that the set S of k gateways is selected. A path p from a sensor node v to a gateway node w is called an *S-routing path* if p contains a gateway node u such that $p[v,u]$ is running outside from S, while $p[u,w]$ is running inside S. In that case, $p[v,u]$ is called the *external prefix* of p, and u is referred as the *S-entrance* of p. If p is equal to its external prefix, then it is called an *external S-routing path*. An *S-routing system* $\mathcal{R}$ is a set of S-routing paths such that each sensor node v is the endpoint of a path in $\mathcal{R}$ with all paths starting from v having the same external prefix. We will call $\mathcal{R}$ a *weak S-routing system* if each path of $\mathcal{R}$ is external. If each sensor node is connected with each gateway node by a path of $\mathcal{R}$, then $\mathcal{R}$ is called a *strong S-routing system*. We will define problems for both weak and strong routing systems with appropriate cost functions. In each case, the cost $c(S)$ of a gateway placement S is the cost of the weak (respectively, strong) S-routing system with minimum cost. In a *Type 1 problem*, a gateway set S is given and determines $c(S)$ and the S-routing system with cost $c(S)$, while in the *Type 2 problem*, the objective is to determine the feasible gateway set S with minimum cost $c(S)$. Generally, *Type 1 problems* can be handled with some standard path algorithms in polynomial time; on the other hand, *Type 2 problems* are hard.

The feasibility of the solutions can be defined in different ways. Regarding the solution, we separate different cases connected to the set of sink nodes that need to know the collected information. The feasibility of the optimization problem depends on the information spread regarding to the sink node level of the network and the user requirements. However, the different cases regarding the feasibility of the problem can be defined in weak and strong routing systems. It is also important to note that different objective functions can be defined regarding the sink placement optimization. In the final system, many factors can be defined by the user, for example, minimizing the largest distance between the sensor and sink nodes, maximizing the number of sink nodes that are reached by the collected data, minimizing the sum of distances, etc. The functions can be used in different ways: a linear combination as a single objective function, bilevel optimization, or multiobjective optimization (e.g., as Pareto optimum).

We conclude the section, for illustrative purposes, with a more concrete solution proposal to the problem. Our network consists of n nodes that we model as vertices V. In the process of designating the sink nodes, we compute the distance matrix $D = [d_{s,d}]$ between all source nodes s and all destination nodes d. In order to define the final optimization problem, for each node $s \in V$, we define the distance to the closest sink node d_s. This yields the definition of the optimization problem, which minimizes:

$$\min_{\text{possible sink placements}} \sum_{s \in V} d_s \quad . \tag{1}$$

In detail, the solution finds k sink nodes or gateways in the following phases:

Phase 1: collect the information about the network at one node;
Phase 2: compute the distance matrix D;
Phase 3: build an MILP (Mixed Integer Linear Program) to solve the optimization problem from Equation (1);
Phase 4: solve MILP;
Phase 5: distribute solution.

The practical implementation of Phase 2: takes $O(n^2 \log^2 n)$ time and the construction of MILP in Phase 3: can also be performed in polynomial time. However, the optimization problem in Equation (1) is, as already mentioned, NP-complete. Therefore, the MILP solution gives us only an approximate solution.

Moreover, the presented solution is centralized and requires further changes, in particular related to parallelization, which, consequently, improves the efficiency and time complexity. For example, replacing MILP with an ant-colony metaheuristics gives a very practical parallel solution [71].

5. Evaluation and Discussion

5.1. Decentralized Data Mining

In the following, we present the experimental setting (Section 5.1.1) and simulation results (Section 5.1.2) of the application of the distributed data mining (DDM) method, presented in Section 4.2, on the air-quality data collected at the “Mrakova Farm” in Bled, Slovenia.

The aim of the simulation is to show a decision tree learned in an incremental way (Hoeffding tree) in a distributed sensor environment does not significantly differ from a decision tree learned in batches from all available data from all of the sensors. Moreover, we want to show that the sequence, in which the nodes are visited in the WSN does not significantly affect the final decision tree.

5.1.1. Experimental Setting

For the purpose of the simulation of our proposed DDM algorithm, we used the air-quality data collected at the “Mrakova Farm” in Bled (Slovenia) during a period of the last half of the year 2020. The data consisted of the air-quality measurements (described in more detail later in this section) from two sensors (one placed on the ground floor and the other on the first floor inside the farmhouse) operating at a 10 s interval from 24 June to 22 December 2020.

We decided to use the data from one sensor only (sensor 1—placed on the ground) because we wanted to avoid dealing with concept drift, since values measured by sensor 1 and those measured by sensor 2 were quite different. The quantities that were measured by the sensors are outlined in Table 1 and include: air temperature (T), relative humidity (RH), CO_2 level (CO_2), air pressure (P), ambient light (AL), and particulate matter particles of size less than 10 μm (PM_{10}).

Table 1. Variables measured by the sensors at “Mrakova Farm”.

Variable	Units	Value Range
T	°C	0–30
RH	%	40–90
CO_2	ppm	400–1700
P	hPa	910–1015
AL	lux	0–50
PM_{10}	μg/m^3	0–2000

Table 1 summarizes the six measured variables, where the first five were chosen as independent ones (attributes), and $PM10$ as the dependent one (class). The task was thus to learn a decision tree that will use the five attributes (T, RH, CO_2, P and AL) to predict the class (PM_{10}).

Our initial dataset, after removing out-of-range measurements (see: Table 1—Value range), contained 951.897 instances—time-stamped sensor measurements of 5 attribute values and 1 class value. We then removed a hold-out set of the last 20% (190.379) of these instances for testing—the test set, leaving in the training set the first 80% (761.518) of instances (a standard k-fold cross-validation evaluation scheme is out of the question here because of the time-series nature of the data). A more elaborate train–test scheme could have been used, but for the purpose of our comparison, it would make no difference.

The experiments were then executed using the WEKA [72] and MOA [65] ML workbenches. Models were learned on the training set and evaluated on the test set. Since we wanted to pursue a classification task, we first discretized the class variable (PM_{10}) into 5 bins using the equal-frequency discretization provided in WEKA—the data were discretized even before splitting them into the train and test sets.

Furthermore, for the purpose of simulating a sensor network, we split the training set into 10 equal subsets, assigning instances i, $i + 10$, $i + 20$, $i + 30$, . . . to the i-th subset (where $i = 1 \ldots 10$). In this way, we simulated collecting the data by 10 instead of just 1 sensor. Because all 10 are just subsets of the same set of measurements of the same phenomenon, there will also be no concept drift, when we later use this data to incrementally learn a decision tree.

5.1.2. Simulation Results

We now first learn a decision tree from the whole training set using the VFDT (Very Fast Decision Tree) [73] implementation of the Hoeffding trees [66] with default parameters in the MOA (Massive On-line Analysis) classifier and evaluate its classification accuracy on the test set. Next, we run the Hoeffding tree learning incrementally by running the same algorithm on the first subset and "augment" this initial tree by successively running the algorithm on all other nine subsets (see Figure 2 in Section 4.2). The final tree's classification accuracy is then again evaluated on the same test set. Since we want to show that the sequence in which the nodes are visited in a network does not significantly affect the final decision tree, we repeat the "incremental" decision tree learning process 10 times, each time on a different random permutation of training set subsets (keeping the first and last subsets in their place). The sequence of this random permutation of the subsets is represented in Table 2.

Table 2. Ten runs of the incremental Hoeffding tree learning process—the subsets sequence.

Run	Subsets Sequence
R1	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9 \to 10$
R2	$1 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9 \to 2 \to 10$
R3	$1 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9 \to 2 \to 3 \to 10$
R4	$1 \to 5 \to 6 \to 7 \to 8 \to 9 \to 2 \to 3 \to 4 \to 10$
R5	$1 \to 6 \to 7 \to 8 \to 9 \to 2 \to 3 \to 4 \to 5 \to 10$
R6	$1 \to 7 \to 8 \to 9 \to 2 \to 3 \to 4 \to 5 \to 6 \to 10$
R7	$1 \to 8 \to 9 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 10$
R8	$1 \to 9 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 10$
R9	$1 \to 6 \to 3 \to 5 \to 4 \to 7 \to 2 \to 9 \to 8 \to 10$
R10	$1 \to 8 \to 4 \to 9 \to 5 \to 6 \to 7 \to 3 \to 2 \to 10$

The results of the test-runs of the Hoeffding tree algorithm are presented in Table 3. The table includes the classification accuracies and sizes (number of leaves in the tree) for each of the 10 runs of the incremental Hoeffding tree (R1. . . R10) + the classification accuracy and size of the Hoeffding tree learned from all the training data (ALL). The average and standard deviation of all classification accuracies and sizes is also provided.

Table 3. Results for 10 runs of the incremental Hoeffding tree learning process—the subsets sequence.

Run	Accuracy (%)	Size (# Leaves)
R1	65.34	285
R2	63.12	291
R3	66.01	280
R4	67.88	279
R5	64.32	285
R6	63.76	290
R7	64.91	281
R8	65.50	292
R9	67.46	291
R10	66.72	285
ALL	64.15	291
Avg.	65.38	286.36
Std.	1.46	4.66

The results in Table 3 show that the trees from different test-runs of the incremental Hoeffding tree algorithm, as well as the Hoeffding tree learned in batch from the complete training set, are very similar regarding classification accuracies and sizes (observe the very low standard deviation of the results in Table 3). This confirms our hypothesis that Hoeffding trees learned in an incremental way do not differ much from their batch counterparts learned from a large "joint dataset". Moreover, the sequence in which the nodes are visited in a WSN does not affect the final incremental Hoeffding tree.

5.2. Security of WSN Data Processing

5.2.1. Privacy Preservation against Eavesdropping

This section analyzes the framework against the attacker model, Eavesdropper (E). The E is interested in learning information about the state of a building equipped with the presented framework by eavesdropping on wireless transmissions generated by nodes of the WSN. There is evidence [15–17] that in the typical WSN scenario where wireless traffic is secured by encryption at data-link-layer or Transport Layer Security analogous, an E could analyze large amounts of WSN traffic to extract descriptive features such as transmission size, occurrence frequency, processing delay, etc. Therefore, the associating of these features with facts or secrets and applying data mining techniques could allow the E to infer over current activities in the monitored region by observing the network traffic.

In the following, we show that the proposed framework withstands traffic analysis attacks sourcing data from the eavesdropping of wireless transmissions of WSN nodes. We motivate the focus on this type of attack by the following three aspects: (a) easy to launch in the indoor monitoring WSN setting; (b) the attack cannot be detected since the attacker is not interfering with the normal functioning of the WSN, and the attack does not require physical access to WSN nodes; and (c) in addition to disclosing privacy, the extracted information could favor other attacks aiming to compromise building security.

The proposed decentralized framework specifies wireless communication involving sink-to-sensor or sensor-to-sensor transmissions. Therefore, the component responsible for ensuring these transmissions' privacy is the DQ3P protocol described in Section 4.3. The DQ3P is a querying protocol in which queries are issued from sink nodes in the form of an onion message. The onion messages are forwarded through a sequence of nodes completing its circuit-like path at the origin sink node. In the following, we recall the terminology defined in Section 4.3.

DQ3P guarantees data privacy against eavesdropping since onion messages are secured by encryption.

DQ3P generates uniform network traffic that does not leak contextual information about activities in the monitored environment due to the following properties:

1. Sensor nodes are not reporting data to the sink node, but the sink node queries sensor nodes.
2. All onion messages traveling the network are of the same size since padding is added at each message processing.
3. DQ3P requires encryption at data-link-layer; therefore, after each message processing, the onion message is forwarded to the next-hop node completely changed by encryption.
4. By DQ3P design, half of the nodes processing an onion message only retain it for a time interval to simulate model learning. We refer to these nodes as decoy nodes.
5. Onion messages travel a randomized path since decoy node addresses are encoded at random positions in the message path during message construction.

Therefore, by property 1, onion message transmission does not imply activity in the monitored environment. By properties, 2, 3, 4, and 5 nodes contributing to model learning cannot be identified by eavesdropping on wireless transmissions. Moreover, if multiple onion messages are issued to the WSN, these properties allow onion messages to mix as they travel through the network. Therefore, even if considering E as able to intercept the whole wireless traffic of the WSN. Due to the mixing of onion messages, it becomes challenging for E even to track the circuit-like onion message path.

5.2.2. Evaluation of DQ3P for Distributed Data Mining

This section presents simulation results of DQ3P, the communication protocol for secure data processing detailed in Section 4.3. The investigation aims to assess the delays introduced by DQ3P applied to train the data mining model described in Section 4.2 on nodes of a WSN.

To conduct the investigation, we base it on the results from the previous study [37]. The previous study provides a scalability study of DQ3P, highlighting that the response times is mainly affected by the onion message size. Therefore, we examine delays introduced by DQ3P by metering the Onion message Round-Trip Time (ORTT) of onion messages having the onion body size corresponding to the size of the data mining model described in Section 4.2, and the onion head of the following number of encryption layers $n = \{5, 20, 60\}$ shown as appropriate in the previous study. In the following, we refer to n as the onion message path length, and we define the ORTT as the elapsed time between the issuing of the onion message from the sink node and the return of the issued onion message to the issuer node. Since by DQ3P design, the processing time of onion messages at WSN nodes is affected by randomness, these delays are not included in the ORTT. Moreover, in the implemented simulation, the measured simulation time does not include processing delays at the application layer by the design of the underlying simulation environment.

Experimental Setup

In order to conduct this investigation, we extended the simulator developed in previous work [37] to allow the broader manipulation of the onion message and the properties of the simulated WSN. The developed simulator is based on the simulation environment nsnam ns-3 [74] and is publicly released and described in [75]. Since the detailed simulation description can be found in [75], in the following we will outline simulator parameters selected to obtain the presented results.

The simulator is set up to construct an ad hoc wireless network of 200 nodes. Nodes are deployed at random locations on a disc-shaped plane of radius 300 m. We selected the deployment scheme affected by randomness to model the usually non-uniform node density of WSNs. The simulated wireless communication conforms to the IEEE 802.11n standard operating at 2.4 GHz at the data rate of 13 Mbps (Modulation Coding Scheme index 1), with the nodes having a wireless communication range of approximately 30 m. The maximum transmission unit and maximum segment size are set to the ns-3 default value, 2296 bytes and 536 bytes, respectively.

Onion messages are transmitted over the TCP protocol, and routing of packets in the multi-hop network is handled using the Optimized Link State Routing Protocol (OLSR) [76]. Encryption layers of the onion head are produced using the *Sealed box*, an ECC-based [77] public-key cipher of 256 b key length implemented in the Libsodium library [78].

As described in [75], the simulator operates in two phases; the first phase is meant to identify nodes in the same network segment as the sink node. Due to the deployment scheme being affected by randomness, some nodes might be located in isolated network segments, unable to communicate with the sink node. In the second phase, the sink node starts issuing onion messages sequentially. After an onion message returns back to the sink node, the following onion message is issued. The sink node is set up to issue 30 onion messages for each value of n. Onion messages are constructed by randomly selecting n nodes to include in the onion message path. The onion message path is encoded in the onion head and the onion body consisting of padding $p = \{1\,\text{k}, 2.5\,\text{k}, 5\,\text{k}, 10\,\text{k}, 25\,\text{k}, 50\,\text{k}, 100\,\text{k}\}$ bytes.

Sensor nodes receiving the onion message are deciphering the outer encryption layer of the onion head to reveal the next-hop IP address and the inner encryption layer. The onion head size is maintained uniform by adding padding of the same number of bytes as the removed onion head layer. The onion body is maintained of uniform size, and the onion message is forwarded to the next-hop node. If the onion message does not reach the next-hop node in 30 s, the onion message is deleted, and a new onion message of the same properties as the interrupted one is issued by the sink node. The 30 s timer is set up to speed up simulation completion.

To select the adequate values of p, we trained a data mining model named *HoeffdingTreeRegressor* [66] from the *skmultiflow* [79] library on preliminary data from the Mrakova farm pilot building. The available data were acquired from 1 sensor, with 15 k measurements about indoor air quality assessment taken during a 2-month interval. The model was trained to estimate the VOC (Volatile Organic Compounds) concentration based on other predictors. We trained the model several times by constraining the maximum model size in bytes to {10 k, 25 k, 50 k}. The high value of the coefficient of determination $0.65 < r^2 < 0.8$ indicates the appropriateness of the values selected for the independent variable p, the onion body size.

Simulation Results

The results of the simulation are displayed in Figure 4, and summary statistics are presented in Table 4. From the collected data, it is possible to notice that at the onion body size of {1 k, 2.5 k, 5 k}, the ORTT measurements do not substantially differ at matching message path length. A probable explanation is the large size of the onion head, which at a message path length of 60 nodes, corresponds to 3127 bytes; therefore, affecting the message transmission more than the onion body at {1 k, 2.5 k}. Moreover, the ORTT is considerably affected by onion body sizes larger than 10 k bytes, and at the onion path length of 60 nodes, the onion body size severely affects the ORTT. Interestingly, from the data in Table 4, the number of interrupted onion messages seems quite irregular. We emphasize that a simulator's parameter affects the number of interrupted onion messages.

In the present investigation, we adopt the ORTT as the indicator of the delays introduced by the DQ3P protocol applied to train a data mining model. The presented results show that onion messages are complete their path on average in less than 1 min at onion body sizes $\leq$ 50 k, even for messages including 60 nodes in the message path. Therefore, delays introduced by DQ3P should be far lower than the cumulative delay introduced by the processing of the data mining model at nodes in the onion message path. Hence, we can conclude that delays introduced by DQ3P applied to train a data mining model of size equal to or smaller to 100 k at onion message path of up to 60 nodes are acceptable.

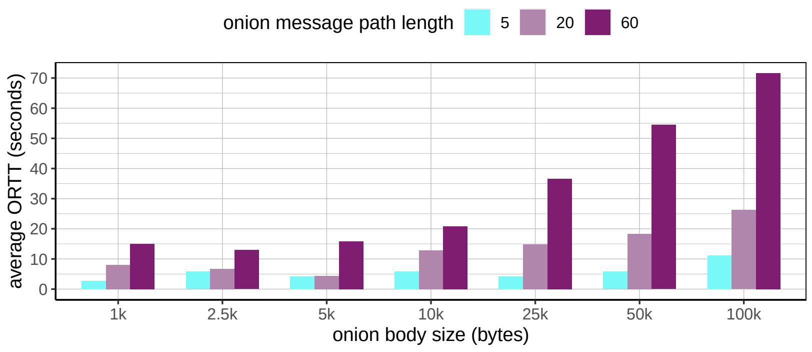

Figure 4. A plot of average ORTT of onion messages traveling through $n = \{5, 20, 60\}$ nodes and having onion body size of p = {1 k, 2.5 k, 5 k, 10 k, 25 k, 50 k, 100 k} bytes. The average was computed on 30 onion messages.

5.3. Decentralized Privacy-Aware Data Storage

In this section, we demonstrate the results of privacy-aware decentralized data storage discussed in Section 4.4.

We used Python 3 to implement the components of our data storage solution. We evaluate the components of our framework using CPU Core i7, 1.80 GHz with 16 GB RAM.

In our motivating scenario, we have five buildings and each building has eight sensors. We show our solution scales up by measuring the timings of our prototype with a growing number of nodes.

Table 4. Summary statistics of onion message ORTT.

Onion Body Size (Bytes)	Message Path Length (Number of Hops)	Onion Head Size (Bytes)	min ORTT (s)	avg ORTT (s)	max ORTT (s)	std ORTT (s)	# of Interrupted Onion Messages
1 k	5	267	0.07	2.66	12.32	3.33	3
	20	1047	0.29	8.07	47.4	9.91	3
	60	3127	1.56	14.97	41.84	12.11	1
2.5 k	5	267	0.09	5.78	26.24	7.67	1
	20	1047	0.46	6.76	21.52	5.83	6
	60	3127	1.64	12.93	36.19	9.87	7
5 k	5	267	0.14	4.22	24.21	7.03	2
	20	1047	0.63	4.34	16.33	4.11	5
	60	3127	5.04	15.85	60.12	11.82	6
10 k	5	267	0.22	5.89	25.39	6.5	4
	20	1047	0.93	12.84	58.81	13.28	3
	60	3127	9.22	20.87	78.34	13.44	2
25 k	5	267	0.37	4.26	23.75	4.96	2
	20	1047	4.42	14.82	43.97	10.22	1
	60	3127	16.45	36.63	72.51	13.46	5
50 k	5	267	0.76	5.78	21.47	4.58	0
	20	1047	6.29	18.29	46.13	11.11	4
	60	3127	30.06	54.44	98.2	17.2	5
100 k	5	267	1.32	11.13	30.42	6.94	1
	20	1047	8.64	26.23	45.94	9.52	2
	60	3127	49.71	71.66	107.16	13.58	8

For the experiments, we observed the time that is needed to write and read data on a decentralized privacy-aware data storage framework. Figure 5 presents time overhead between different numbers of sensors such as 8, 16, 24, 32, and 40. We calculated the average time consumption in seconds for data read and write operations.

In the case of 8 sensors, the time to write data has an average of 0.420 s, and read data have an average of 0.042 s. For the case of 16 sensors, the data write time has an average of 0.451 s that does not show much difference with the case of 8 sensors. Data read time has an average of 0.058 s that is slightly higher than the average read time of 8 sensors. For the case of 24 sensors, the average data write time is 0.552 s, and the average data read time is 0.054 s that is slightly less than the read time of 16 sensors.

In the case of 32 sensors, the average time to write data is 0.665 s, and the data read time is 0.065 s. For 40 sensors, the data write gives an average of 0.689 s, and the data read has an average of 0.082 s. The results show that a privacy-aware data storage prototype gives a reasonable time overhead. The average time to write and read data is not affected much by increasing the number of sensors. We explain that this result comes from the multiple optimizations in the Python interpreter and at the operating system level (system cache).

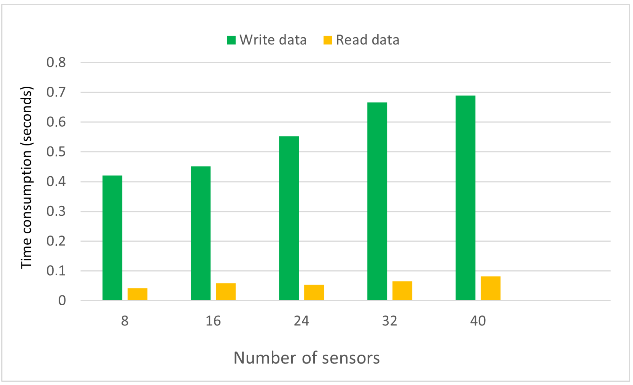

Figure 5. Overview of overall time needed for read and write operations on different numbers of sensors.

5.4. Routing and Sink Placement Optimization

The dominating set and facility location problems provide a reliable theoretical frame for our sink node placement optimization problem. Nevertheless, our approach is to combine the toolset, restrictions, and different objective functions of the given theoretical frame in the area of infection models. The advantage of using infection models (with appropriate modifications) is that it can guarantee proper theoretical bounds regarding the solution of the optimization problem as well as a reliable theoretical environment for our problem. An example for the modification of the infection models can be found in [80]. In our previous work, we introduced an infection model based on a simulation environment using the greedy method with different spreading strategies and objectives for Wireless Sensor Networks [81]. Due to the restrictions discussed in the given paper, our initial work provided a solution for the centralized version of a sensor network. Regarding the methodology used, it is important to note that, due to the submodularity of the problem we introduced, the method in the area of infection models performs within a constant factor of $1 - 1/e$ as good as the optimal solution. The basic problem was further investigated in our latest research, which is, however, a generalized solution that can be used in a decentralized solution [82]. The main problem of the greedy algorithm is that it provides an insufficient running time in the case of big networks, and it is not efficiently scalable. In our preliminary results, we were aiming to reduce the running time of the method while, at the same time, keeping the quality of the solution. Using structural properties of the given network, we were able to improve the runtime as well as provide an environment to benchmark and evaluate different community detection methods. Our solution places the seed nodes (or, in a sink placement problem, the sink nodes), in places that are somehow in a central position between dense subnetworks identifying the critical positions that can endanger the wireless sensor network from a connectivity point of view. In the paper, we have compared our solution using eight different methods with centrality-based approaches, as well as with the original greedy method. As a final result, we can say that our methodology is efficient from two points of view. First, it is suitable to reduce the search space of the general optimization problem (as well as the sink node placement optimization), and second, it provides an environment to test and compare the effectiveness of different methods in reducing the search space of the sink placement optimization problem. The research gives us a good basic model with which we are able to evaluate the different solutions regarding to the structural properties of such networks.

6. Conclusions

In this paper, we present an integrated edge computing solution for decentralized Indoor Air Quality monitoring. Our solution features the following advantages that address well-known limitations of Wireless Sensors Networks: distributed storage using Distributed Hash Table, blockchain-based Role-based Access Control for data privacy, onion routing for monitoring protection, routing and sink placement optimization, and distributed data mining.

In particular, this paper features the following original contributions: (1) a decentralized data storage solution combining DHT and blockchain with a novel metadata structure that supports data updates and privacy management, combined with a flexible solution for supporting different security mechanism; (2) a novel protocol based on onion routing that supports decentralized data mining while preserving privacy against monitoring; and (3) the introduction of a theoretical concept called S-routing system that provides insights on possible developments for routing optimization. We prove the feasibility of our solution with indoor air quality monitoring prototypes and simulations to be deployed in different pilots. Our solution executes distributed data mining for IAQ data and produces results while remaining fully decentralized and including security, privacy, and routing optimization.

Future work includes the integration and real-life deployment of our solution in large scale environments and studying its applicability to different use cases that require different data processing or different quantities of data. Further development includes an adaptable solution that, based on the data and on the type of analysis required from the use case, automatically adjusts to the most adapted choices for network optimization, data storage and analysis, privacy and security. Agent-based computing and autonomic computing are inspirations towards such a line of work.

Author Contributions: Writing—review & editing, M.M., A.T., N.H., S.A., B.D., L.H., M.K., A.B. and B.K. All authors have read and agreed to the published version of the manuscript.

Funding: The authors gratefully acknowledge the European Commission for funding the InnoRenew project (Grant Agreement #739574) under the Horizon2020 Widespread-Teaming program and the Republic of Slovenia (Investment funding of the Republic of Slovenia and the European Regional Development Fund). They also acknowledge the Slovenian Research Agency ARRS for funding the project J2-2504.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors acknowledge Marina Paldauf for her contribution to the development of the data mining part.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Tham, K.W. Indoor air quality and its effects on humans—A review of challenges and developments in the last 30 years. *Energy Build.* **2016**, *130*, 637–650. [CrossRef]
2. Kumar, P.; Martani, C.; Morawska, L.; Norford, L.; Choudhary, R.; Bell, M.; Leach, M. Indoor air quality and energy management through real-time sensing in commercial buildings. *Energy Build.* **2016**, *111*, 145–153. [CrossRef]
3. Garcia Lopez, P.; Montresor, A.; Epema, D.; Datta, A.; Higashino, T.; Iamnitchi, A.; Barcellos, M.; Felber, P.; Riviere, E. *Edge-Centric Computing: Vision and Challenges*; ACM SIGCOMM: Amsterdam, The Netherlands, 2015.
4. Saini, J.; Dutta, M.; Marques, G. Indoor Air Quality Monitoring Systems Based on Internet of Things: A Systematic Review. *Int. J. Environ. Res. Public Health* **2020**, *17*, 4942. [CrossRef] [PubMed]
5. Abreu, A.; Lopes, S.I.; Manso, V.; Curado, A. Low-Cost LoRa-Based IoT Edge Device for Indoor Air Quality Management in Schools. In *Science and Technologies for Smart Cities*; Paiva, S., Lopes, S.I., Zitouni, R., Gupta, N., Lopes, S.F., Yonezawa, T., Eds.; Springer: Cham, Switzerland, 2021; pp. 246–258.
6. Cocos, H.N.; Merkl, D. Decentralized Data Processing On The Edge—Accessing Wireless Sensor Networks with Edge Computing. In Proceedings of the 16th International Conference on Applied Computing, Cagliari, Italy, 15–16 September 2019; pp. 265–269.

7. Sun, S.; Zheng, X.; Villalba-Díez, J.; Ordieres-Meré, J. Indoor Air-Quality Data-Monitoring System: Long-Term Monitoring Benefits. *Sensors* **2019**, *19*, 4157. [CrossRef] [PubMed]
8. Zheng, X.; Lu, J.; Sun, S.; Kiritsis, D. Decentralized industrial IoT data management based on blockchain and IPFS. In *IFIP International Conference on Advances in Production Management Systems*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 222–229.
9. Legault, M. A Practitioner's View on Distributed Storage Systems: Overview, Challenges and Potential Solutions. *Technol. Innov. Manag. Rev.* **2021**, *11*, 32–41. [CrossRef]
10. Benedict, S.; Rumaise, P.; Kaur, J. IoT blockchain solution for air quality monitoring in SmartCities. In Proceedings of the 2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Goa, India, 16–19 December 2019; pp. 1–6.
11. Shih, D.H.; Shih, P.Y.; Wu, T.W. An infrastructure of multi-pollutant air quality deterioration early warning system in spark platform. In Proceedings of the 2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), Chengdu, China, 20–22 April 2018; pp. 648–652.
12. Estrin, D.; Govindan, R.; Heidemann, J.; Kumar, S. Next Century Challenges: Scalable Coordination in Sensor Networks. In Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, MobiCom '99, Seattle, WA, USA, 15–20 August 1999; Association for Computing Machinery: New York, NY, USA, 1999; pp. 263–270. [CrossRef]
13. Gan, W.; Lin, J.C.W.; Chao, H.C.; Zhan, J. Data mining in distributed environment: A survey. *WIREs Data Min. Knowl. Discov.* **2017**, *7*, e1216. [CrossRef]
14. Laube, P.; Duckham, M. Decentralized Spatial Data Mining for Geosensor Networks. In *Geographic Data Mining and Knowledge Discovery*; Routledge: London, UK, 2008. [CrossRef]
15. Gu, T.; Fang, Z.; Abhishek, A.; Mohapatra, P. IoTSpy: Uncovering Human Privacy Leakage in IoT Networks via Mining Wireless Context. In Proceedings of the 2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications, London, UK, 31 August–3 September 2020; pp. 1–7.
16. Zhang, F.; He, W.; Liu, X. Defending against traffic analysis in wireless networks through traffic reshaping. In Proceedings of the 2011 31st International Conference on Distributed Computing Systems, Minneapolis, MI, USA, 20–24 June 2011; pp. 593–602.
17. Saltaformaggio, B.; Choi, H.; Johnson, K.; Kwon, Y.; Zhang, Q.; Zhang, X.; Xu, D.; Qian, J. Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic. In Proceedings of the 10th {USENIX} Workshop on Offensive Technologies ({WOOT} 16), Austin, TX, USA, 10–11 August 2014.
18. Xu, J.; Yang, G.; Chen, Z.; Wang, Q. A survey on the privacy-preserving data aggregation in wireless sensor networks. *China Commun.* **2015**, *12*, 162–180. [CrossRef]
19. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 223–238.
20. Wilson, R.L.; Rosen, P.A. Protecting data through perturbation techniques: The impact on knowledge discovery in databases. *J. Database Manag.* **2003**, *14*, 14–26. [CrossRef]
21. Conti, M.; Zhang, L.; Roy, S.; Di Pietro, R.; Jajodia, S.; Mancini, L.V. Privacy-preserving robust data aggregation in wireless sensor networks. *Secur. Commun. Netw.* **2009**, *2*, 195–213. [CrossRef]
22. He, W.; Liu, X.; Nguyen, H.; Nahrstedt, K.; Abdelzaher, T. Pda: Privacy-preserving data aggregation in wireless sensor networks. In Proceedings of the IEEE INFOCOM 2007–26th IEEE International Conference on Computer Communications, Anchorage, AK, USA, 1–12 May 2007; pp. 2045–2053.
23. He, W.; Nguyen, H.; Liuy, X.; Nahrstedt, K.; Abdelzaher, T. iPDA: An integrity-protecting private data aggregation scheme for wireless sensor networks. In Proceedings of the MILCOM 2008 IEEE Military Communications Conference, San Diego, CA, USA, 16–19 November 2008; pp. 1–7.
24. Bista, R.; Yoo, H.K.; Chang, J.W. A new sensitive data aggregation scheme for protecting integrity in wireless sensor networks. In Proceedings of the 2010 10th IEEE International Conference on Computer and Information Technology, Bradford, UK, 29 June–1 July 2010; pp. 2463–2470.
25. Blaß, E.O.; Zitterbart, M. An efficient key establishment scheme for secure aggregating sensor networks. In Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, Taipei, Taiwan, 21–24 March 2006; pp. 303–310.
26. Westhoff, D.; Girao, J.; Acharya, M. Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution, and routing adaptation. *IEEE Trans. Mob. Comput.* **2006**, *5*, 1417–1431. [CrossRef]
27. Castelluccia, C.; Mykletun, E.; Tsudik, G. Efficient aggregation of encrypted data in wireless sensor networks. In Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, San Diego, CA, USA, 17–21 July 2005; pp. 109–117.
28. Lin, Y.H.; Chang, S.Y.; Sun, H.M. CDAMA: Concealed data aggregation scheme for multiple applications in wireless sensor networks. *IEEE Trans. Knowl. Data Eng.* **2012**, *25*, 1471–1483. [CrossRef]
29. Othman, S.B.; Bahattab, A.A.; Trad, A.; Youssef, H. Confidentiality and integrity for data aggregation in WSN using homomorphic encryption. *Wirel. Pers. Commun.* **2015**, *80*, 867–889. [CrossRef]
30. Hayouni, H.; Hamdi, M.; Kim, T.H. A survey on encryption schemes in wireless sensor networks. In Proceedings of the 2014 7th International Conference on Advanced Software Engineering and Its Applications, Hainan Island, China, 20–23 December 2014; pp. 39–43.

31. Zhao, C.; Zhao, S.; Zhao, M.; Chen, Z.; Gao, C.Z.; Li, H.; Tan, Y.A. Secure multi-party computation: Theory, practice and applications. *Inf. Sci.* **2019**, *476*, 357–372. [CrossRef]
32. Vaidya, J.; Clifton, C. Secure set intersection cardinality with application to association rule mining. *J. Comput. Secur.* **2005**, *13*, 593–622. [CrossRef]
33. Kantarcioglu, M.; Clifton, C. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 1026–1037. [CrossRef]
34. Du, W.; Zhan, Z. Building Decision Tree Classifier on Private Data. In Proceedings of the IEEE International Conference on Privacy, Security and Data Mining–Volume 14, Maebashi City, Japan, 1 December 2002; pp. 1–8.
35. Cock, M.d.; Dowsley, R.; Nascimento, A.C.; Newman, S.C. Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data. In Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security, Denver, CO, USA, 8 June 2015; pp. 3–14.
36. Jung, T.; Mao, X.; Li, X.Y.; Tang, S.J.; Gong, W.; Zhang, L. Privacy-preserving data aggregation without secure channel: Multivariate polynomial evaluation. In Proceedings of the 2013 IEEE INFOCOM, Turin, Italy, 14–19 April 2013; pp. 2634–2642.
37. Hrovatin, N.; Tošić, A.; Mrissa, M.; Vičič, J. A General Purpose Data and Query Privacy Preserving Protocol for Wireless Sensor Networks. *arXiv* **2021**, arXiv:cs.CR/2111.14994.
38. Longo, F.; Nicoletti, L.; Padovano, A.; d'Atri, G.; Forte, M. Blockchain-enabled supply chain: An experimental study. *Comput. Ind. Eng.* **2019**, *136*, 57–69. [CrossRef]
39. Chakravorty, A.; Rong, C. Ushare: User controlled social media based on blockchain. In Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, Beppu, Japan, 5–7 January 2017; pp. 1–6.
40. Huang, H.; Zhou, X.; Liu, J. Food supply chain traceability scheme based on blockchain and EPC technology. In *International Conference on Smart Blockchain*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 32–42.
41. Zyskind, G.; Nathan, O. Decentralizing privacy: Using blockchain to protect personal data. In Proceedings of the 2015 IEEE Security and Privacy Workshops, San Jose, CA, USA, 21–22 May 2015; pp. 180–184.
42. Shafagh, H.; Burkhalter, L.; Hithnawi, A.; Duquennoy, S. Towards blockchain-based auditable storage and sharing of IoT data. In Proceedings of the 2017 on Cloud Computing Security Workshop, Dallas, TX, USA, 3 November 2017; pp. 45–50.
43. Ali, S.; Wang, G.; White, B.; Cottrell, R.L. A blockchain-based decentralized data storage and access framework for pinger. In Proceedings of the 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), New York, NY, USA, 1–3 August 2018; pp. 1303–1308.
44. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2001.
45. Hedrick, C. *Routing Information Protocol*; RFC 1058; Internet Engineering Task Force: Fremont, CA, USA, 1988.
46. Malkin, G. *RIP Version 2*; RFC 2453; Internet Engineering Task Force: Fremont, CA, USA, 1998.
47. Malkin, G.; Minnear, R. *RIPng for IPv6*; RFC 2080; Internet Engineering Task Force: Fremont, CA, USA, 1997.
48. Brodnik, A.; Grgurovič, M.; Požar, R. Modifications of the Floyd-Warshall algorithm with nearly quadratic expected-time. *Ars Math. Contemp.* **2021**, *22*. [CrossRef]
49. Zhang, L.Y.; Jian, M.; Li, K.P. A Parallel Floyd-Warshall algorithm based on TBB. In Proceedings of the 2010 2nd IEEE International Conference on Information Management and Engineering, Kunming, China, 26–28 November 2010; pp. 429–433. [CrossRef]
50. Yu, J.; Chen, Y.; Ma, L.; Cheng, X. On Connected Target k-Coverage in Heterogeneous Wireless Sensor Networks. *Sensors* **2016**, *16*, 104. [CrossRef]
51. Bhushan, M.; Narasimhan, S.; Rengaswamy, R. Robust sensor network design for fault diagnosis. *Comput. Chem. Eng.* **2008**, *32*, 1067–1084. [CrossRef]
52. Zorbas, D.; Raveneau, P.; Ghamri-Doudane, Y. Assessing the cost of deploying and maintaining indoor wireless sensor networks with RF-power harvesting properties. *Pervasive Mob. Comput.* **2017**, *43*, 64–77. [CrossRef]
53. Engmann, F.; Katsriku, F.; Abdulai, J.D.; Adu-Manu, K.; Banaseka, F. Prolonging the Lifetime of Wireless Sensor Networks: A Review of Current Techniques. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 1–23. [CrossRef]
54. Haynes, T.W.; Hedetniemi, S.T.; Slater, P.J. *Domination in Graphs: Advanced Topics*; Routledge: London, UK, 1998.
55. Zanjirani Farahani, R.; Hekmatfar, M. *Facility Location: Concepts, Models, Algorithms and Case Studies*; Springer: Berlin/Heidelberg, Germany, 2009.
56. Frendrup, A.; Tuza, Z.; Dahl, P. *Distance Domination in Vertex Partitioned Graphs*; Research Report Series No. R-2009-10; Department of Mathematical Sciences, Aalborg University: Aalborg, Denmark, 2009.
57. Zhang, Z.; Liu, Q.; Li, D. Two Algorithms for Connected R-Hop k-Dominating Set. *Discret. Math. Algorithms Appl.* **2009**, *1*, 485–498. [CrossRef]
58. Santos Coelho, R.; Moura, P.; Wakabayashi, Y. The k-hop connected dominating set problem: Hardness and polyhedra. *Electron. Notes Discret. Math.* **2015**, *50*, 59–64. [CrossRef]
59. Nguyen, M.; Hà, M.; Nguyen, D.; Tran, T. Solving the k-dominating set problem on very large-scale networks. *Comput. Soc. Netw.* **2020**, *7*, 1–5. [CrossRef]
60. Shang, W.; Wan, P.J.; Yao, F.; Hu, X. Algorithms for minimum -connected -tuple dominating set problem. *Theor. Comput. Sci.* **2007**, *381*, 241–247. [CrossRef]

61. Sheng, H.; Du, D.; Sun, Y.; Sun, J.; Zhang, X. Approximation Algorithm for Stochastic Set Cover Problem. In Proceedings of the Algorithmic Aspects in Information and Management, 14th International Conference, AAIM 2020, Jinhua, China, 10–12 August 2020; pp. 37–48.
62. Eisenbrand, F.; Grandoni, F.; Rothvoß, T.; Schaefer, G. Approximating connected facility location problems via random facility sampling and core detouring. In Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, USA, 20–22 January 2008.
63. Swamy, C.; Kumar, A. Primal–Dual Algorithms for Connected Facility Location Problems. *Algorithmica* **2004**, *40*, 245–269. [CrossRef]
64. Bandyapadhyay, S.; Roy, A.B. Approximate Covering with Lower and Upper Bounds via LP Rounding. *arXiv* **2020**, arXiv:abs/2007.11476.
65. Bifet, A.; Gavald, R.; Holmes, G.; Pfahringer, B. *Machine Learning for Data Streams: With Practical Examples in MOA*; The MIT Press: Cambridge, MA, USA, 2018.
66. Domingos, P.; Hulten, G. Mining High-Speed Data Streams. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA, USA, 20–23 August 2000; Association for Computing Machinery: New York, NY, USA, 2000; pp. 71–80. [CrossRef]
67. Syverson, P.F.; Goldschlag, D.M.; Reed, M.G. Anonymous connections and onion routing. In Proceedings of the 1997 IEEE Symposium on Security and Privacy (Cat. No. 97CB36097), Oakland, CA, USA, 4–7 May 1997; pp. 44–54.
68. Aslam, S.; Tošić, A.; Mrissa, M. Secure and Privacy-Aware Blockchain Design: Requirements, Challenges and Solutions. *J. Cybersecur. Priv.* **2021**, *1*, 164–194. [CrossRef]
69. Bertino, E. RBAC models—Concepts and trends. *Comput. Secur.* **2003**, *22*, 511–514. [CrossRef]
70. Radi, M.; Dezfouli, B.; Abu Bakar, K.; Lee, M. Multipath Routing in Wireless Sensor Networks: Survey and Research Challenges. *Sensors* **2012**, *12*, 650–685. [CrossRef] [PubMed]
71. Brodnik, A.; Grgurovič, M. Parallelization of Ant System for GPU under the PRAM Model. *Comput. Inform.* **2018**, *37*, 229–243. [CrossRef]
72. Witten, I.H.; Frank, E.; Hall, M.A.; Pal, C.J. *Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques*, 4th ed.; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2016.
73. Hulten, G.; Spencer, L.; Domingos, P. Mining time-changing data streams. In Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 26–29 August 2001; pp. 97–106.
74. Henderson, T.R.; Lacage, M.; Riley, G.F.; Dowell, C.; Kopena, J. Network simulations with the ns-3 simulator. *SIGCOMM Demonstr.* **2008**, *14*, 527.
75. Hrovatin, N.; Tošić, A.; Vičič, J. PPWSim: Privacy Preserving Wireless Sensor Network Simulator. 2021. Available online: https://ssrn.com/abstract=3978796 (accessed on 20 December 2021).
76. Clausen, T.; Jacquet, P.; Adjih, C.; Laouiti, A.; Minet, P.; Muhlethaler, P.; Qayyum, A.; Viennot, L. *Optimized Link State Routing Protocol (OLSR)*; RFC3626; INRIA: Cedex, France, 2003.
77. Miller, V.S. Use of elliptic curves in cryptography. In *Conference on the Theory and Application of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1985; pp. 417–426.
78. Libsodium The Sodium Crypto Library. Available online: https://libsodium.gitbook.io/doc/ (accessed on 28 May 2021).
79. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
80. Cicalese, F.; Cordasco, G.; Gargano, L.; Milanič, M.; Peters, J.G.; Vaccaro, U. Spread of influence in weighted networks under time and budget constraints. *Theor. Comput. Sci.* **2015**, *586*, 40–58. [CrossRef]
81. Hajdu, L.; Dávid, B.; Krész, M. Gateway placement and traffic load simulation in sensor networks. *Pollack Period.* **2021**, *16*, 102–108. [CrossRef]
82. Hajdu, L.; Krész, M.; Bóta, A. Evaluating the role of community detection in improving influence maximization heuristics. *Soc. Netw. Anal. Min.* **2021**, *11*, 1–11. [CrossRef]