applied sciences

MDPI

Article

Privacy-Preserving Data Mining on Blockchain-Based WSNs

Niki Hrovatin [1,2,†], Aleksandar Tošić [1,2,†], Michael Mrissa [1,2,†] and Branko Kavšek [1,3,*,†]

1 Faculty of Mathematics, Natural Sciences and Information Technologies, University of Primorska, Glagoljaška 8, 6000 Koper, Slovenia; niki.hrovatin@famnit.upr.si (N.H.); aleksandar.tosic@upr.si (A.T.); michael.mrissa@innorenew.eu (M.M.)
2 InnoRenew CoE, Livade 6a, 6310 Izola, Slovenia
3 Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
* Correspondence: branko.kavsek@upr.si
† These authors contributed equally to this work.

Abstract: Currently, the computational power present in the sensors forming a wireless sensor network (WSN) allows for implementing most of the data processing and analysis directly on the sensors in a decentralized way. This shift in paradigm introduces a shift in the privacy and security problems that need to be addressed. While a decentralized implementation avoids the single point of failure problem that typically applies to centralized approaches, it is subject to other threats, such as external monitoring, and new challenges, such as the complexity of providing decentralized implementations for data mining algorithms. In this paper, we present a solution for privacy-aware distributed data mining on wireless sensor networks. Our solution uses a permissioned blockchain to avoid a single point of failure in the system. Contracts are used to construct an onion-like structure encompassing the Hoeffding trees and a route. The onion-routed query conceals the network identity of the sensors from external adversaries, and obfuscates the actual computation to hide it from internally compromised nodes. We validate our solution on a use case related to an air quality-monitoring sensor network. We compare the quality of our model against traditional models to support the feasibility and viability of the solution.

Keywords: WSN; air quality; privacy; blockchain; data mining

check for updates

Citation: Hrovatin, N.; Tošić, A.; Mrissa, M.; Kavšek, B. Privacy-Preserving Data Mining on Blockchain-Based WSNs. *Appl. Sci.* **2022**, *12*, 5646. https://doi.org/10.3390/app12115646

Academic Editor: Federico Divina

Received: 25 April 2022
Accepted: 31 May 2022
Published: 2 June 2022

1. Introduction

Currently, wireless sensor networks (WSNs) are widely used in various application domains (e.g., healthcare, supply chains, or agriculture). They support collecting and storing data about the state of monitored objects (e.g., patients, products, or crops), enabling further analysis that benefits stakeholders (e.g., better health, traceability, increased productivity). As typical WSNs rely on cloud infrastructures to handle the large amount of collected data, they are subject to diverse issues related to maintenance, scalability, and vulnerability to security threats. Most cloud-related issues originate from its centralized design, naturally opening the way to decentralized systems as an alternative to handle data. In addition, the recent development of embedded technology available in sensors is a strong incentive for the design of decentralized solutions. Present-day smart sensors have such strong computational capacities that they can handle demanding tasks, including hosting an entire operating system and the full web protocol stack, including a web server, or they are directly connected to one. Therefore, they become part of the web, supporting the vision of a worldwide sensor web [1]. Such a context makes the decentralized operation of the data collection and management processes very appealing.

Indeed, decentralized systems have seen quite a strong development since the release of Bitcoin, a practical blockchain implementation that provides decentralized trust, leading to the development of various other implementations. While, for centralized WSNs, security attacks target specific weak points of the system, in decentralized WSNs, attacks consist in observing the flow of messages that go through the network, and in identifying the role of each node to better target further attacks. Additionally, the data mining tasks performed in centralized databases, usually in the cloud, need some adjustments to be realized in a decentralized WSN.

Therefore, there is a major need to (1) rethink data mining tasks so that they better fit the decentralized underlying system where they should be executed, and (2) protect the privacy of these tasks with moderate additional computing requirements. Several trends of work exist in this direction; however, most are very computationally intensive, or show high communication costs.

In this paper, we present a solution for privacy-preserving DM (PPDM) that relies on the blockchain to provide trust, and on onion routing (OR) to preserve the privacy of the DM tasks that are executed in WSNs.

In particular, our contribution features smart contracts to provide for role-based access control and secure DM execution , so that nodes can trust each other. Our solution, instead of sharing data between nodes, is to share the partially constructed models so that each sensor node only has access to the partial model that is the result of the computation of the previous sensor nodes.

The remainder of this paper is structured as follows: In Section 2, we overview the most relevant work on PPDM, OR, and blockchain, and highlight the research gap which our work covers. In Section 3, we present our architecture and detail how its different components are articulated. In Section 4, we detail the methodology followed to enable its operation over a use case related to indoor air quality monitoring. In Section 5, we present the results obtained and compare them to the closest work in the literature. Finally, we summarize our main contributions and present some guidelines for future work in Section 6.

2. Related Work

In this section, the related work relevant to our research is presented, and knowledge gaps are identified. Section 2.1 gives an overview of the state of the art in privacy-preserving data mining in WSNs, Section 2.2 presents the recent research on the use of onion routing in WSNs, and Section 2.3 lists the relevant applications of the blockchain on WSNs.

2.1. Privacy-Preserving Data Mining in WSNs

Conventionally, DM on WSN data is realized in a centralized processing point external to the WSN. While it is simple to implement, such an approach raises privacy concerns due to the required relocation of sensor node data to the external system through the WSN, a multi-hop wireless network operated by resource-constrained devices. There are many studies addressing privacy concerns in WSNs [2–4], several of them indicating that, in addition to data privacy, contextual privacy must also be considered.

A considerable amount of literature has been published on privacy-preserving data aggregation since it is particularly suited to WSNs' characteristics [5,6]. The concept is based on aggregating data as it travels through the multi-hop network towards the sink node, and on applying privacy-preservation techniques during aggregation points to prevent the disclosure of the data of individual sensor nodes. Although the data aggregate does not disclose individual sensor privacy, aggregating the data may also reduce the knowledge obtainable using DM.

Local differential privacy (LDP) is well established in DM applications that require the privacy of individual data sources [7]. Data is encoded or perturbed at the origin or intermediate point, and then sent to the consumer. Therefore, LDP techniques also leverage the trade-off between privacy and data utility. The authors in [8] discuss the application of LDP in decentralized IoT networks. An LDP IoT framework is presented in [9]; however, LDP is applied only at edge servers.

Homomorphic encryption was proposed for preserving privacy, both in DM [10] and WSN applications [6]. Privacy is assured since computations are performed on cryptographically secured data without decryption. However, the technique is computationally intensive [11,12], and reasonable solutions for WSNs are designed only to compute addition and multiplication on encrypted data.

Secure multi-party computation (SMC) [13] allows multiple parties to jointly compute a function over their inputs while keeping those inputs private. Many studies [10,14] use the SMC technique to preserve the privacy of individual data sources during DM operations. However, traditional solutions relying on SMC require high computation and communication costs; thus, they are not convenient for application in WSNs. Therefore, SMC in WSNs is limited to low-complexity tasks, such as data aggregation [15].

Data mining algorithms that are performed over a WSN are inherently distributed, which has shifted the focus from traditional data mining (DM) approaches to the paradigm of distributed data mining (DDM) in recent years. The authors of [16] provide a relatively recent survey of state-of-the-art DDM techniques, focusing also on privacy-preserving distributed data mining (PPDDM). On the other hand, [17] 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. This is especially relevant for our research, where we aim at monitoring indoor air quality over a WSN.

2.2. Onion Routing in WSNs

Onion routing has a long tradition in enabling anonymous communication in public networks, first using layered objects to establish anonymous connections [18], and now being implemented in the TOR network to construct anonymous connections one hop at a time using a key-agreement protocol [19].

Although combined applications in the context of PPDM are rare [20], onion routing remains a privileged technique to protect from external monitoring attacks, as is proven by existing work on the topic [21].

De Cristofaro et al. [22] used a layered object to retrieve data from a WSN without disclosing the identity of the queried sensor node. Although the technique preserves privacy effectively, it only allows for data retrieval from individual nodes. One important work related to query privacy over WSNs is the work of Carbunar et al. [23], which proposes a similar solution, although with different underlying architectural choices (i.e., no region-based servers and a different trust model). The solution works by using source routing to privately route a declarative query to the aggregator nodes of a WSN. However, aggregator nodes are a point of failure for the network since they collect data from the sensor nodes in their proximity.

In general, we noted that most related work adopts a strong separation between data collection/querying and data mining. This highlights the originality of our work, where the data remains on the sensors, with a direct decentralized implementation of the data mining process that involves using onion routing for the purpose of protecting the tasks together with the processed data.

2.3. Blockchain Applications in WSNs

The unique properties that blockchain networks have, and how they can benefit IoT and WSNs, has been studied extensively. Early research focused on using a blockchain network as an external component of systems that are mainly used as databases for key storage and WSN management [24,25]. Blockchain networks have been used in combination with WSNs for data security, sensor node authentication [26], removing single points of failure in WSNs [27], and secure data accumulation [28].

Recent advances in blockchain protocols have enabled the design of light clients, which are nodes participating in the network without needing to maintain the entire chain. This enables clients to run on sensor nodes with low computing power and storage requirements. Coupled with smart contract platforms, new applications, such as malicious node detection [29] and anomaly detection [30], have surfaced.

Although we use blockchain for access control, which is a typical way to ensure trust between actors in a decentralized setup, the originality of our work lies in the use of smart blockchain contracts to ensure the authenticity of the onion messages, which is a novel approach to blockchain when combined with onion routing, as in this context.

2.4. Main Contributions

1. We develop a fully decentralized PPDDM framework for building incremental models where data never leaves the sensor, further improving data privacy and security;
2. By using onion routing, we are able to obfuscate computations between participating sensors and external adversaries;
3. We remove any SPOF by implementing RBAC and key storage using blockchain;
4. We show that the PPDDM approach achieves comparable accuracy to traditional DM approaches for the air quality monitoring use case.

3. Architecture

Our architecture is built over a set of sensor nodes that have the computing power of a Raspberry Pi, and is connected to an IAQ sensor that collects temperature (in Celsius), relative humidity, dew point, absolute humidity, CO_2, VOC index, and luminance.

Each sensor node gathers local data from its attached sensor, and at the same time, is a blockchain node. The proposed architecture requires a blockchain supporting smart contract functionality. The term "smart contract" was first defined in [31] as a computerized transaction protocol that executes the term of a contract. Today, the term "smart contract" is adopted for software that is recorded on the blockchain and that is executed by nodes maintaining the blockchain as part of transaction processing. Ethereum [32] was the first blockchain to implement Turing-complete smart contracts. This was achieved by introducing the Ethereum Virtual Machine (EVM), the gas concept, and the account-based transaction model. In addition to allowing the encoding of arbitrary state transition functions, the smart contract functionality implemented in the Ethereum network allows the smart contract to maintain a state; therefore, more complex applications can be built.

The OpenEthereum private network could serve our system as a secure, decentralized, and immutable smart contract platform. The permissioned Ethereum network operates the proof of authority (PoA) [33] consensus; therefore, the network is maintained by a selected set of validator nodes. Other nodes do not contribute to the state transition processing; therefore, these nodes can run an Ethereum light client [34].

An illustration of the system architecture is given in Figure 1. The use of onion routing for PPDDM is described in Section 3.1. The operation of the proposed architecture and a smart contract description is given in Section 3.2.

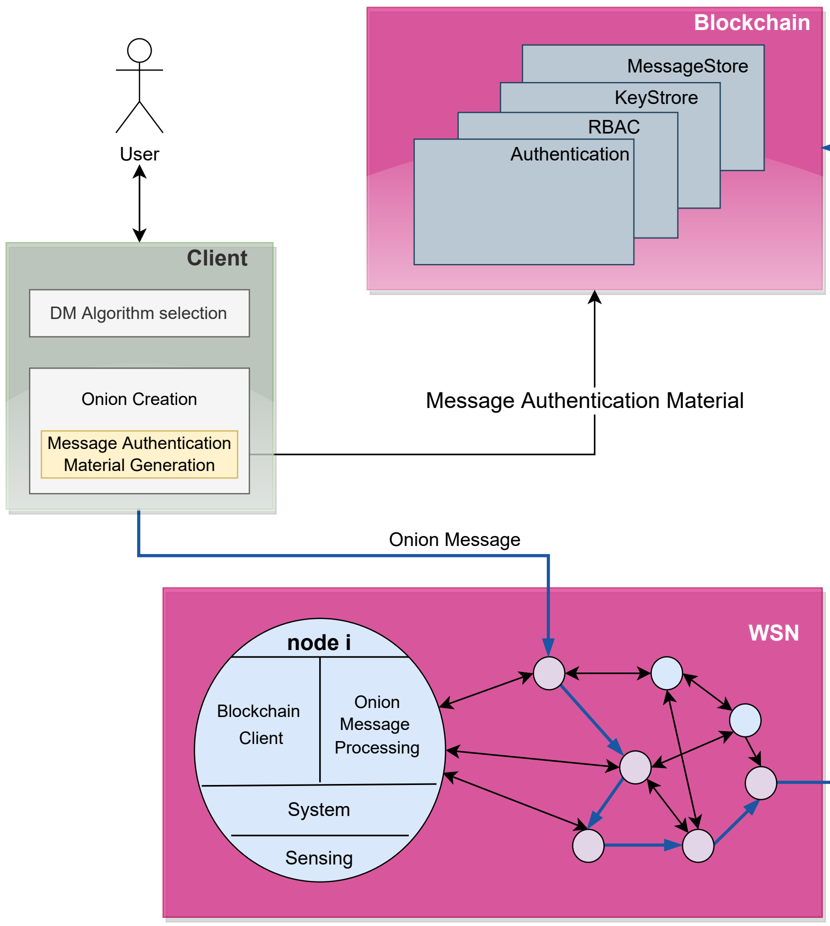

Figure 1. The figure displays the architecture of the proposed system and the interaction between the user, the WSN, and the blockchain.

3.1. Onion Routing for Privacy-Preserving Decentralized Data Mining

This section describes how to use the onion routing technique for privacy-preserving decentralized data mining. The onion routing technique builds upon messages consisting of encryption layers ciphered using public-key cryptography. Each layer of a message contains the necessary encryption key material and the address of the node that possesses the secret key, which will decipher the next layer. Therefore, the layered object is deciphered following the path given at message construction and delivers encryption key material to in-path systems. The encryption key material is used to establish a connection between the layered object origin and the last receiver. The connection is anonymous since none of the systems involved in the communication will know the whole connection route, apart from the layered object origin.

To achieve privacy-preservation for DDM, we rely on a new use of onion routing that was first described in [35]. Similar to onion routing, this technique uses messages consisting of several encryption layers, and each layer includes path details. However, the encryption key material is not included in each layer of the layered object, and it is not used to establish an anonymous connection. Instead, the encryption key material gives access to the payload accompanying the layered object. Therefore, only specific nodes in the message path are able to access the message payload, and their identity is concealed among the set of nodes routing the layered object.

Our system uses this technique to convey a DM model to WSN nodes. Figure 2 illustrates a message in our system. The DM model is enclosed in the payload accompanying the layered object. Nodes receiving encryption key material from the layered object perform model processing on their local data. Therefore, model processing is realized on WSN nodes without a central processing point. The technique ensures data privacy since the private sensed data of nodes does not leave the origin node. The generated communication traffic is uniform; nodes performing model processing are concealed among the set of nodes routing the message, the message path and sojourn time are randomized, the message size is uniform, and the messages are indistinguishable by encryption and are mixed when routed in the network. Therefore, external actors eavesdropping on a wireless communication cannot learn information from the network traffic.

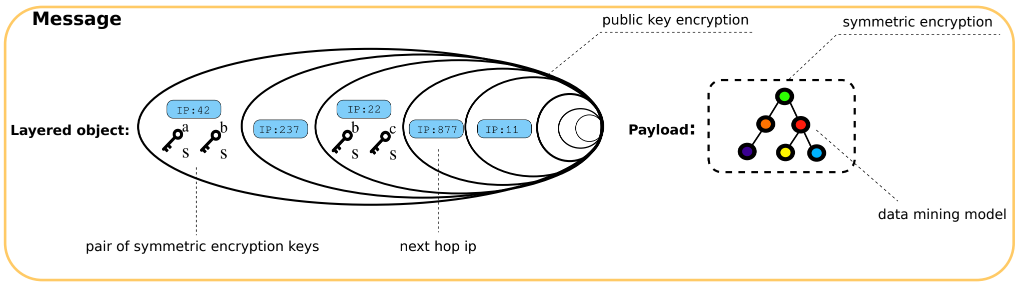

Figure 2. A message of the privacy-preserving protocol described in [35], adapted for DM model processing. The layers of the layered object, containing pairs of symmetric encryption keys, include distinct keys. The first key is used for payload decryption, and the second key for payload encryption after DM model processing. Please note that encryption key pairs are chained through layers of the layered object.

3.2. Role-Based Access Control and Secure Data Mining with Smart Contract

The smart contract enabling our architecture supports the following functionalities:

- *Authentication:* allows sensor node authentication and stores the sensor node public key in the *KeyStore*;
- *Role-Based Access Control (RBAC):* verifies user permissions. The smart contract contains the roles assigned to the public keys of authorized users. Roles enforce the set of DM algorithms that a user can execute on nodes of the WSN;
- *Query Submission:* authenticates the user with RBAC and writes the message authentication material in the *MessageStore*;
- *Message Authentication:* a view function called by sensor nodes to verify the message origin and permissions using the message authentication material;
- *Result Submission:* allows authenticated sensor nodes to submit the message result and remove the corresponding message authentication material from the *MessageStore*.

The outlined architecture allows for the management of user roles with RBAC functionalities. A user entrusted with a specific role is allowed to construct a message made of the layered object and the payload, as is described in Section 3.1. The payload content is constrained by the user role, as defined by the RBAC.

The user interrogates the *KeyStore* to learn sensor node addresses for message construction. After message construction, the user calls the *query submission* functionality providing the message authentication material. The message authentication material consists of a Merkle proof [36] constructed from the layered object layers, and the signature of the Merkle proof produced using the user's private key. The user then feeds the constructed message to the WSN.

Each sensor node receiving a message performs the following tasks. The layer of the layered object that is dedicated to this specific node is deciphered, and the message is authenticated by calling the *Message Authentication* function. The message authentication is performed by verifying the hash of the layered object's content against the Merkle proof in the *MessageStore*. The user's role and permission for DM model execution are also verified; this is conducted using the signature attached to the Merkle proof.

The penultimate layer of the layered object holds a flag signaling its position. The sensor node detecting this flag calls the *Result Submission* function and stores the message result. For simplicity, we assume that the message result is stored on the blockchain; however, the result could be stored in a separate data store, submitting on-chain only the message authentication material, the hash of the result, and the resource locator.

The user that issued the request tracks the chain for results associated with the message authentication material of the query submission. The obtained message result consists of the last layer of the layered object and the accompanying payload enclosing the DM model. Next, the user decrypts the layered object with its private key, revealing the symmetric encryption key that gives access to the payload content.

Please note that problems related to Byzantine faults are out of the scope of our work, so we assume in this paper that the nodes collect the data properly, and execute exactly the tasks assigned to them.

4. Methodology

This section presents the methodology behind the experimental evaluation of our proposed PPDDM method in detail. Section 4.1 gives an outline of the dataset that was used in the experiments—the GAMS IAQ dataset [37]. In Section 4.2, the "traditional" data mining algorithms that were used for comparison against our PPDDM are presented. Finally, Section 4.3 provides all the details about our PPDDM method and describes how the experimental evaluation was performed.

4.1. Indoor Air Quality Dataset

In order to validate the proposed distributed data mining approach, we look at the GAMS indoor air quality (IAQ) dataset, which is publicly available in [37]. This dataset was used for experimenting and benchmarking in several research works [38–40]. In this work, we use the GAMS dataset to compare the distributed data mining approach with traditional centralized data mining.

As reported in Table 1, the GAMS dataset includes observations from six IAQ indicators. Observations were collected from 21 November 2016 to 28 March 2017 on a minute basis. From further dataset inspection, we identified that, for benchmarking, it could be interesting to estimate the CO_2 level from T, RH, and VOC.

Table 1. The IAQ indicators of the GAMS dataset.

Feature	Sample Value	Unit
Carbon dioxide (CO_2)	708.0	ppm
Relative humidity (RH)	40.09	%
Particulate matter 2.5 micron ($PM_{2.5}$)	10.2	$\mu g/m^3$
Particulate matter 10 micron (PM_{10})	9.0	$\mu g/m^3$
Temperature (T)	20.83	Celsius
Volatile organic compounds (VOC)	0.093	ppm

The data were pre-processed by averaging observations on a six-minute interval. Table 2 summarizes the statistics of the pre-processed GAMS dataset.

Table 2. Summary of the statistics of the pre-processed GAMS dataset.

	CO_2	T	RH	VOC
mean	712.1	23.00	28.40	0.1168
std	405.61	2.08	5.46	0.086
min	370.2	18.04	22.05	0.062
25%	431.70	21.43	34.70	0.0635
50%	494.0	22.89	38.30	0.0740
75%	900.1	24.73	42.05	0.1405
max	2603.0	27.95	72.09	0.8702

All measured features include 29,420 observations.

4.2. Traditional Data Mining

Here, by "traditional" data mining, we mean machine learning (ML) algorithms run in a centralized fashion on data that is available in memory. The data used to learn and evaluate the "traditional" ML algorithms were the same pre-processed GAMS IAQ data with summary statistics that are presented in Table 2.

For the ML algorithms selected in this phase of the experimental evaluation, we used the WEKA ML framework [41]—an open-source data mining (DM) framework developed in the University of Waikato in New Zealand, containing over 200 implementations of various ML and DM algorithms.

Since all the attributes that describe our GAMS dataset are of numeric type, it was natural to select regression algorithms for the comparisons. Three well-known regression algorithms were selected, namely, linear regression, model trees (the M5′ algorithm [42,43]), and random forest [44]. The linear regression algorithm was selected as a baseline, because it is the most well-known and studied algorithm; the M5′ model trees algorithm was selected because it can produce more sophisticated, yet still well-explainable models, and the Random Forest was selected with the hope of producing a highly accurate result.

All three ML algorithms were run with their default parameters on the first 90% of the GAMS IAQ dataset to learn a model, which was then evaluated on the remaining 10% of the data. This rather unintuitive evaluation protocol was used (instead of the standard k-fold cross validation) because of the time-series nature of the GAMS dataset.

4.3. Distributed Data Mining

The distributed data mining setting described in Section 3 characterizes a network of nodes that store the last historical data about locally sensed IAQ features. A DM algorithm is then privately conveyed from node to node and executed in situ. Therefore, the DM model cannot access the global data—only the data provided by the node on which the DM model is executing is accessible to the model. Moreover, conveying large DM models from node to node in a WSN is resource-demanding and significantly affects the response times of the proposed architecture. Therefore, it is necessary to send the least amount of data.

To validate the distributed data mining approach, we first partition the GAMS IAQ dataset to simulate the described context according to Section 4.3.1. Section 4.3.3 gives the selected DDM algorithm for comparison with traditional centralized DM and outlines the DDM algorithm processing on the partitioned data. Section 4.3.4 details the experiment design for assessing the message propagation time at different DM model sizes.

4.3.1. Data Partitioning

The GAMS pre-processed dataset was partitioned into ten subsets to resemble a WSN of 10 closely located nodes. As shown in Figure 3, we applied round-robin partitioning. Therefore, each subset holds data from the entire monitored period, and data is chronologically ordered. As can be read from Figure 3, each observation summarizes a 1-hour interval of a sensor node.

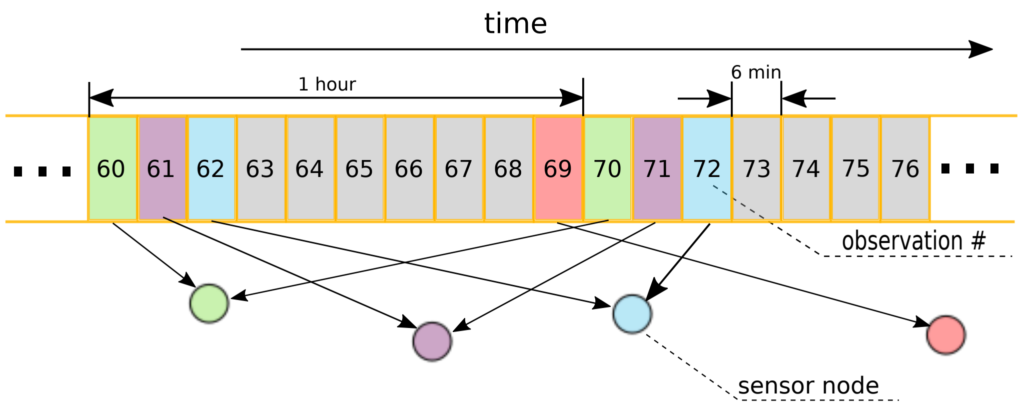

Figure 3. Data partitioning according to the round-robin method.

4.3.2. Design of the Experiment

Each sensor node collects its own data (in our simulation, there is actually a round-robin data assignment scheme that simulates sensor node data collection, as is depicted in Figure 3). The sensor node data is then used to "augment" the Hoeffding tree received from the previous sensor node, producing an updated Hoeffding tree that is then passed to the next sensor node. The whole procedure is depicted in Figure 4 and has already been proved effective in [45].

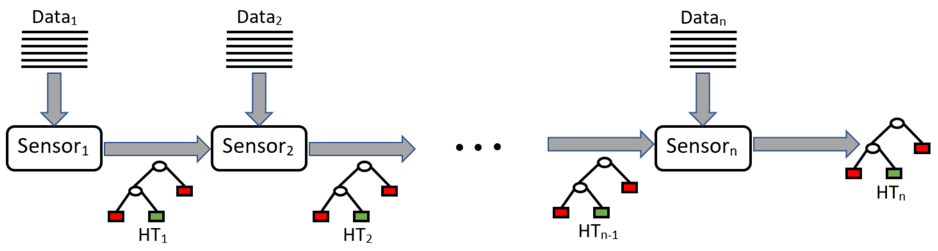

Figure 4. Incremental learning with Hoeffding trees.

As can be observed from Figure 4, the initial Hoeffding tree (HT_1) is first learned in the batch from the data provided by the first sensor node—$Sensor_1$ ($Data_1$). 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 Hoeffding tree (HT_{k-1}) and the data from the current sensor node—$Sensor_k$ ($Data_k$). The final Hoeffding tree is then represented by the last tree in this chain—HT_n.

Such a partial Hoeffding tree's (HT_k's) propagation through the WSN (instead of the actual sensor data being propagated) is most suitable for preserving privacy, and also diminishes the load on the WSN.

4.3.3. Distributed Data Mining Algorithm

Due to the context of distributed data, where only limited local data is available for processing, we cannot run the same algorithms as in Section 4.2. Instead, we look at the Hoeffding tree [46], a very fast decision tree algorithm proposed for stream data. The algorithm builds a decision tree as data arrives, without the need to reuse instances from already-processed data. This is achieved by maintaining the statistics needed for splitting attributes in each leaf node and determining when a split should occur as data arrives, using the Hoeffding's bound [47]. The Hoeffding probability bound enables us to compute ε, which can be used as a confidence interval for the estimation of the split.

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta))}{2N}}$$

R is the range of the random variable r. By performing N-independent observations of the variable r, and computing their mean $\bar{r}$, the Hoeffding bound enables us to state with confidence $1 - \delta$ that $\bar{r}$ is within ε distance from the true mean of the variable r.

In [46], it was shown that the Hoeffding tree output is asymptotically nearly identical to that of a traditional batch learner in infinitely many examples.

We use the Python implementation provided in the scikit-multiflow [48] library, specifically, the regression variant named *HoeffdingTreeRegressor*. Development of the scikit-multiflow library stopped in the year 2020, when the developers of scikit-multiflow and Creme [49], another Python library for data stream mining, decided to merge their efforts and release a new library named River [50]. Even though the state of the art of data stream mining is currently implemented in the River library, we decided to employ the *HoeffdingTreeRegressor* implemented in the scikit-multiflow since this implementation allowed us to control the tree size in bytes. The *HoeffdingTreeRegressor* algorithm at the base of the implementation in River is identical to the one in scikit-multiflow. The algorithm is based on the Fast and Incremental Model Trees (FIMT-DD) method described in [51], without concept drift adaptation.

The pseudo-code is given in Algorithm 1. From the code, it is possible to notice that the algorithm starts with an empty tree that keeps statistics at the leaves from arriving data. Each time k values arrive at a leaf, the algorithm finds the best split for each attribute and ranks the attributes according to the standard deviation reduction (SDR) measure [51]. Therefore, attributes are ranked such that the standard deviation is maximally reduced, if splitting by the highest ranked attribute, by considering the ratio of the SDR values between any of the attributes and the highest-ranked attribute as the random variable r with range $R \in [0, 1]$. If the inequality $\bar{r} + \varepsilon < 1$ holds, then the highest-ranked attribute is the best split over the whole distribution for that portion of data.

Algorithm 1: Pseudo-code of the *HoeffdingTreeRegressor*.

```
1  Begin with an empty leaf (root)
2  repeat
3      Read the next example;
4      Traverse the example to a leaf;
5      Update statistics in the leaf;
6      if k examples were read then
7          Find best split per attribute;
8          Rank attributes using the SDR measure;
9          if splitting criterion is satisfied then
10             Make a split on the best attribute;
11             make two new branches leading to (empty) leaves;
12         end
13     end
14 until End of the stream;
```

We ran the ML algorithm with the sample average leaf predictor for the following maximal model sizes: $M = \{5k, 10k, 25k, 50k, 100k, 250k, 500k, 1M, 2.5M, 5M\}$ (bytes). In the provided code snippet (Listing 1), the `MAX_TREE_SIZE` parameter on line 8 corresponds to one of the values of M. The algorithm was run on the partitioned pre-processed GAMS IAQ dataset. To closely simulate the architecture described in Section 3, we design the experiment as if model processing occurs once a day, processing 24 h of data from each node following the round-robin approach. Therefore, the *HoeffdingTreeRegressor* processes 24 h of data from subset #0, then moves to data from subset #1, and so on. After processing at subset #9, the model moves to the next-day data from subset #0 and repeats until it reaches the end of data.

Listing 1. Hoeffding Tree training and evaluation.

```
1  from skmultiflow.trees import HoeffdingTreeRegressor
2  data = pd.read_csv("gams_aq/gams_preprocessed.csv")
3  X = data[['temperature', 'humidity', 'voc']]
4  Y = data['co2']
5  y_actual = []
6  y_predicted = []
7
8  model = HoeffdingTreeRegressor(max_byte_size=MAX_TREE_SIZE, leaf_prediction="mean")
9
10 for i in range(len(Y)):
11 prediction = model.predict(X[i])
12 y_predicted.append(prediction)
13 y_actual.append(Y[i])
14 model.partial_fit(X[i], Y[i])
```

We say that the *HoeffdingTreeRegressor* is processing data because when a data instance is provided to the algorithm, it first estimates the response variable value and then uses the observed value for model learning. These operations are shown in the provided code snippet (Listing 1) on lines 11 and 14, respectively. To hold consistency with Section 4.2, in Section 5, we report the model evaluation results on the last 10% of the data.

4.3.4. Message Propagation Time

The system we are presenting makes use of the onion routing technique described in Section 3.1 to convey a DM model through the nodes of a WSN. Therefore, messages are routed through several nodes of a wireless multi-hop network and relayed multiple times, introducing substantial latency before receiving the result. To assess the response time of the proposed architecture operating over a large WSN, we look at the publicly released NS3-based simulator described in [52]. NS3 [53] is a discrete-event network simulator for internet systems, and it is the most-used network simulator by the IoT research community [54]. The simulator described in [52] is designed to simulate the technique described in Section 3.1 over a WSN that varies in node number, topology, routing, etc. Furthermore, it allows for the manipulation of message structures by defining nodes in the message path (corresponding to layers of the layered object), detailing the payload content, and other settings such as maintaining message uniformity.

We set up a simulation that constructs a WSN of 200 nodes. Nodes are deployed at random locations on a disc-shaped plane with a radius of 300 m. We selected the deployment scheme affected by randomness to model the usually non-uniform node density of WSNs. Nodes have a wireless communication range of approximately 30 m, operating based on the IEEE 802.11n standard at 2.4 GHz and with a data rate of 13 Mbps. The maximum transmission unit and maximum segment size are set to the ns3 default value, namely, 2296 bytes and 536 bytes, respectively.

Messages are transmitted over the TCP protocol, multi-hop routing is conducted using the Optimized Link State Routing Protocol (OLSR) [55], and cryptography is completed with the Libsodium library [56].

The simulator is set up to issue 20 messages for each value of M, with M being the size in bytes of the model carried in the message payload. The layered object included in the message is constructed to lead the message through 20 randomly selected nodes. Our previous work [35] stated that for privacy requirements, half the nodes in the message path are only performing routing operations, without access to the payload content (decoy nodes). Therefore, this setup allows for estimations of the propagation time of messages constructed to perform model processing at 10 nodes. Message size is maintained as uniform after the onion routing operation, with padding when necessary. Figure 5 shows the simulated WSN and an onion message routed through 20 sensor nodes.

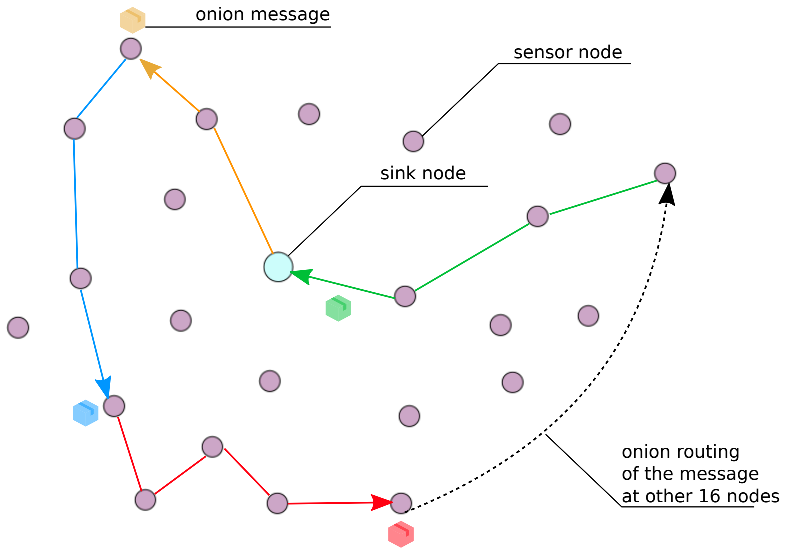

Figure 5. Example of the disc-shaped WSN with random node placement generated in the simulator [52]. The sink node is deployed in the center of the network, issuing onion messages sequentially. Each onion message is onion-routed at 20 nodes and ends its path at the sink node. In the figure, we indicate an onion routing operation with the change in color of the packet.

5. Results

5.1. Distributed Data Mining Approach

To validate our approach, we analyze the accuracy of Hoeffding trees in predicting CO_2, depending on the set tree size. Figure 6 shows the accuracy of predicted CO_2 compared to observed values from sensors, depending on the tree size. The last 10% of the chronologically ordered data was used in the evaluation phase. We observe that the overall accuracy of Hoeffding trees improves with the increase in tree size. As discussed in Section 4.3.3, Hoeffding trees store attribute statistics in leaves; therefore, as new leaves are formed due to split decisions, the model size increases, and also the prediction accuracy should increase.

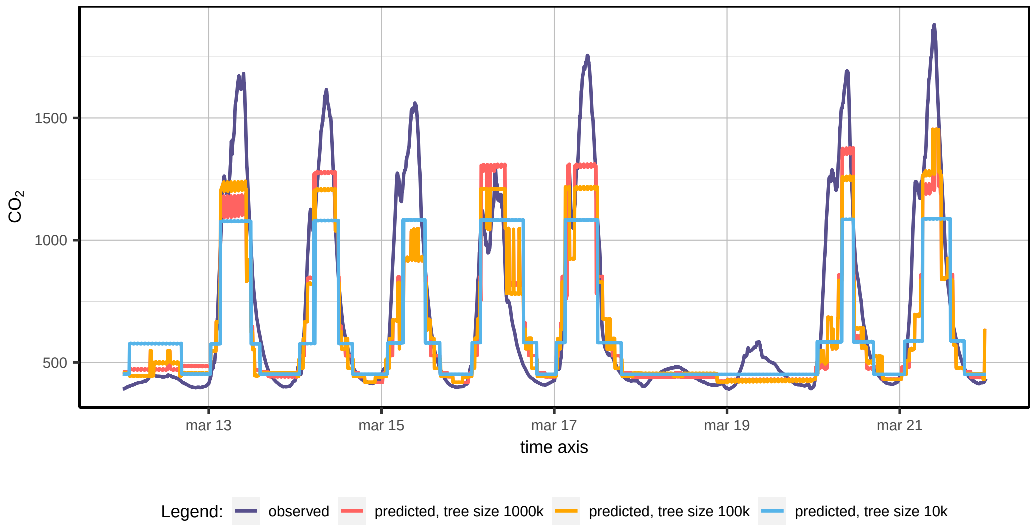

Figure 6. Comparison between predicted CO_2 using Hoeffding trees and observed for different tree sizes.

While training Hoeffding trees, we make continuous predictions in order to monitor the progressive improvements in accuracy through time. Figure 7 illustrates four different tree sizes and their respective prediction accuracies. The box plots show the aggregated weekly data points for clarity. We observe that, in all cases, the variance and mean decrease with time. Moreover, we observe that a small tree size (5 Kb) is worse overall when compared to the other sizes, while increasing the size of the tree beyond 100 Kb has no marginal effect on the accuracy.

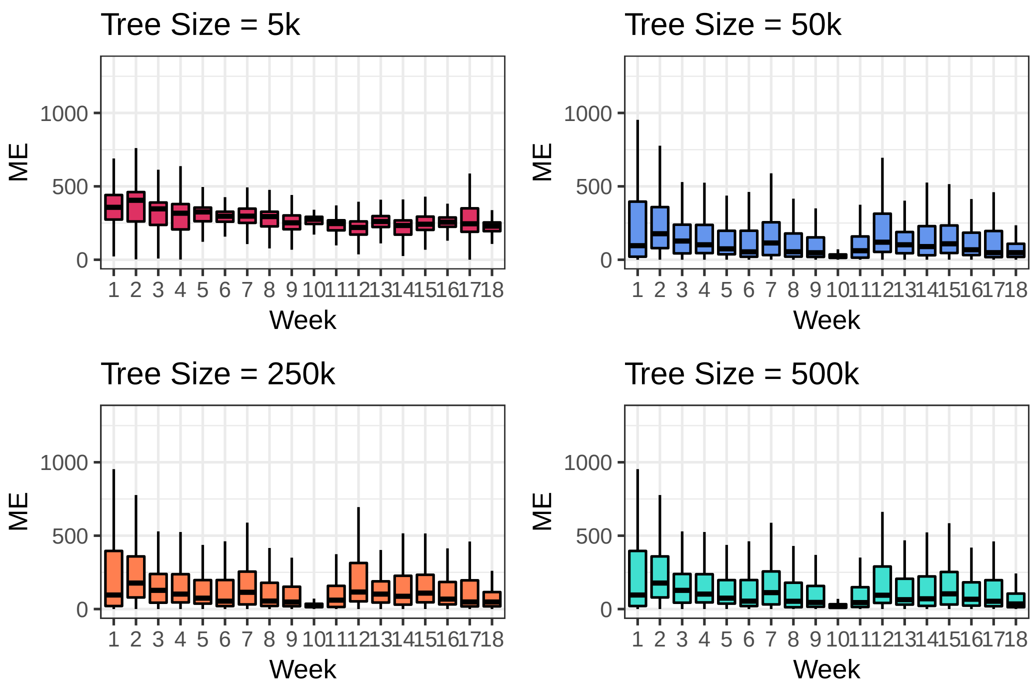

Figure 7. Overall prediction accuracy of different-sized Hoeffding trees while learning.

Figure 8 shows the comparison between the Hoeffding tree and the traditional DM algorithms. We compared the absolute error between the predicted and observed values on a daily basis. All models were trained using the first 90% of the data, and evaluated using the last 10%. We observe that the prediction error of the Hoeffding tree is comparable with traditional algorithms throughout all the reported days in Figure 8. Additionally, we provide standard model quality metrics in Table 3, obtained from a model evaluation on the last 10% of the data. From Figure 8, it is possible to notice that the Hoeffding tree model outperforms the random forest and M5′ models. This can be also read in Table 3. From Table 3, it appears that the linear regression model is the best predictor, achieving the highest values for the RMSE, RRSE, and COR metrics. Although the Hoeffding tree model did not achieve the best results, the quality metrics reported in Table 3 and Figure 8 confirm that the DDM approach using the Hoeffding tree achieves comparable accuracy to traditional DM approaches. Furthermore, our DDM approach can be compared to the research in [40], where researchers developed an inference model that was also tested on the same GAMS IAQ dataset [37]. They report a coefficient of determination of 0.6742 when predicting the CO_2 value from other predictors, whereas deriving the coefficient of determination from Table 3 for the Hoeffding tree results in a value of 0.7109.

Table 3. Statistical comparison of the prediction accuracy of different models. MAE—mean absolute error, RMSE—root mean squared error, RRSE—root relative squared error, and COR—correlation coefficient.

Algorithm	MAE	RMSE	RRSE	COR
Random forest	154.8616	249.2222	61.5162	0.803
M5′	148.5085	232.2563	57.3284	0.8323
Linear regression	157.7538	212.1302	52.3606	0.8717
Hoeffding tree	135.3154	230.9903	57.0625	0.8432

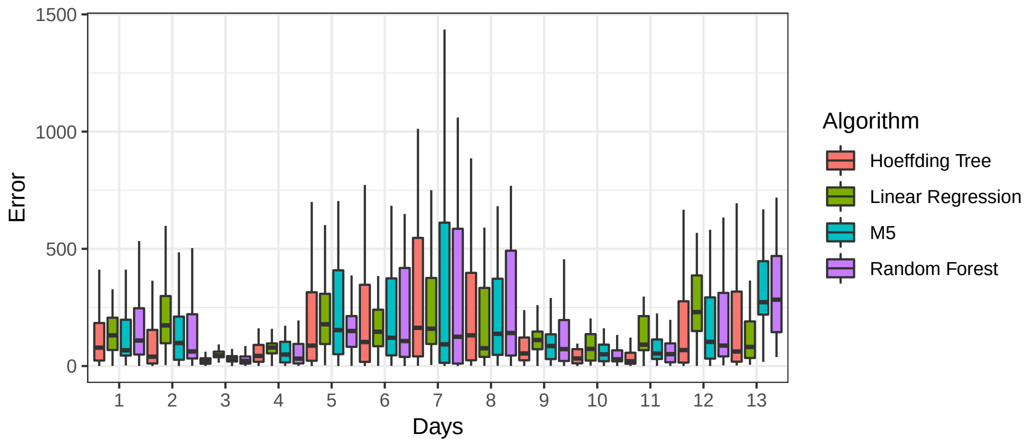

Figure 8. Absolute error comparison between the Hoeffding tree and traditional algorithms. The box plots aggregate the daily absolute error per DM model.

5.2. Validation of the Privacy-Preserving Architecture

The described privacy-preserving architecture builds upon the Ethereum blockchain running the PoA consensus and the privacy-preserving communication protocol described in [35]. Since the use of the blockchain in our architecture does not diverge from the typical use, we focus on evaluating the privacy-preserving communication protocol for distributed data mining. Moreover, the adoption of blockchain technology for resource-constrained devices was discussed in several other research works; specifically, we point out [57], which provides performance analyses of the Ethereum blockchain running the PoA consensus in IoT networks.

The privacy-preserving communication protocol uses the onion routing technique, which builds on messages having several layers of public-key cryptography; public-key cryptography is known for being computationally intensive. However, in the proposed architecture, multiple public-key cryptography operations are required only at the point of message creation; therefore, this is conducted by the user on the client, which is supposed to be running on a typical consumer machine. Nodes of the WSN at the onion message receipt perform only one public-key cryptography operation. To assess the time required for a node of our system to decipher a message, we coded a test script that measures the required time for ciphering several bytes of data using the same cipher as the simulator [52]. The cipher is based on the elliptic curve Curve2551 [58], and is implemented in the Libsodium library [56] under the name *Sealed Box*. By our measurments, encrypting 1 kB and 16 kB of data using the *Sealed Box* implementation on a Raspberry Pi4 takes, respectively, 603 and 735 microseconds. Therefore, the processing time required for cryptographic operations is low compared to the DM model processing and the message propagation time. On the other hand, DM model processing depends on the amount of data to be processed at each node and on the computational complexity of the applied DM model. In the proposed architecture, we describe the use of the DM model known as Hoeffding tree, which is known to be memory and computationally efficient [59].

To show the feasibility of the onion-routed approach, we study the impact of message propagation through the WSN based on different tree sizes. The experiment was designed to simulate the worst-case scenario with random node deployment, inducing a high packet drop rate. Figure 9 shows the average propagation time of an onion message relative to the size of the Hoeffding tree. We observe a low impact of larger trees on encryption, decryption, and training on total propagation times.

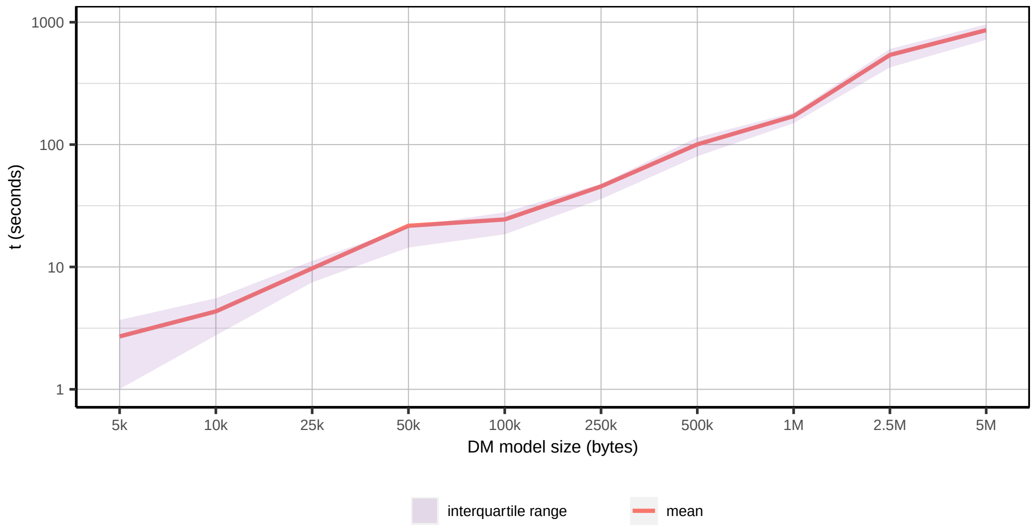

Figure 9. The average propagation time of onion messages through 20 WSN nodes, conveying a DM model of size M. To provide a better overview of the collected data, we report the log transformation of the y axis.

6. Conclusions

In this paper, we describe an innovative combined use of blockchain and onion routing to realize privacy-preserving decentralized data mining over wireless sensor networks. The advantage of such an approach is to avoid problems such as single points of failure and dependencies on remote infrastructures, while providing qualitatively similar results to those of traditional data mining techniques.

In effect, our solution relies on smart contracts and onion routing to address further problems that arise from the decentralized nature of the solution. Onion routing protects the decentralized data mining process from external monitoring, and smart contracts introduce a way to trust the computation being realized over the different sensors that form the network. We rely on a public key infrastructure to provide for data integrity and confidentiality.

Future work includes exploring advanced methodologies for data processing and data storage to guarantee properties such as fault tolerance or resilience against network partition. Pushing the boundaries of data mining on the edge of the network opens doors for smart, autonomous, embedded systems to support human activities, while reducing the required environmental impact.

Author Contributions: Conceptualization, N.H. and A.T.; Data curation, N.H., A.T. and B.K.; Formal analysis, M.M.; Funding acquisition, M.M.; Investigation, N.H., A.T., M.M. and B.K.; Methodology, N.H., A.T. and B.K.; Project administration, A.T. and M.M.; Resources, M.M. and B.K.; Software, N.H.; Supervision, A.T. and M.M.; Validation, N.H., A.T. and B.K.; Visualization, N.H. and A.T.; Writing—original draft, N.H., A.T., M.M. and B.K.; Writing—review & editing, N.H., A.T., M.M. 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.

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. Gibbons, P.; Karp, B.; Ke, Y.; Nath, S.; Seshan, S. IrisNet: An architecture for a worldwide sensor Web. *IEEE Pervasive Comput.* **2003**, *2*, 22–33. [CrossRef]
2. Li, N.; Zhang, N.; Das, S.K.; Thuraisingham, B. Privacy preservation in wireless sensor networks: A state-of-the-art survey. *Ad Hoc Netw.* **2009**, *7*, 1501–1514. [CrossRef]
3. Tomić, I.; McCann, J.A. A survey of potential security issues in existing wireless sensor network protocols. *IEEE Internet Things J.* **2017**, *4*, 1910–1923. [CrossRef]
4. Jiang, J.; Han, G.; Wang, H.; Guizani, M. A survey on location privacy protection in wireless sensor networks. *J. Netw. Comput. Appl.* **2019**, *125*, 93–114. [CrossRef]
5. Bista, R.; Chang, J.W. Privacy-preserving data aggregation protocols for wireless sensor networks: A survey. *Sensors* **2010**, *10*, 4577–4601. [CrossRef] [PubMed]
6. 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]
7. Kasiviswanathan, S.P.; Lee, H.K.; Nissim, K.; Raskhodnikova, S.; Smith, A. What can we learn privately? *SIAM J. Comput.* **2011**, *40*, 793–826. [CrossRef]
8. Sun, M.; Tay, W.P. On the relationship between inference and data privacy in decentralized IoT networks. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 852–866. [CrossRef]
9. Xu, C.; Ren, J.; Zhang, D.; Zhang, Y. Distilling at the edge: A local differential privacy obfuscation framework for IoT data analytics. *IEEE Commun. Mag.* **2018**, *56*, 20–25. [CrossRef]
10. Mendes, R.; Vilela, J.P. Privacy-preserving data mining: Methods, metrics, and applications. *IEEE Access* **2017**, *5*, 10562–10582. [CrossRef]
11. 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]
12. 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, Haikou, China, 20–23 December 2014; pp. 39–43.
13. 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]
14. 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, 16 October 2015; pp. 3–14.
15. 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 Proceedings IEEE INFOCOM, Turin, Italy, 14–19 April 2013; pp. 2634–2642.
16. 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]
17. Laube, P.; Duckham, M. Decentralized Spatial Data Mining for Geosensor Networks. In *Geographic Data Mining and Knowledge Discovery*; Routledge: London, UK, 2008. [CrossRef]
18. Reed, M.G.; Syverson, P.F.; Goldschlag, D.M. Anonymous connections and onion routing. *IEEE J. Sel. Areas Commun.* **1998**, *16*, 482–494. [CrossRef]
19. Dingledine, R.; Mathewson, N.; Syverson, P. *Tor: The Second-Generation Onion Router*; Technical report; Naval Research Lab.: Washington, DC, USA, 2004.
20. El Mougy, A.; Sameh, S. Preserving Privacy in Wireless Sensor Networks using Onion Routing. In Proceedings of the 2018 International Symposium on Networks, Computers and Communications (ISNCC), Rome, Italy, 19–21 June 2018; pp. 1–6. [CrossRef]
21. Ravi, N.; Jeyanthi, P. Secure Inter Hop Verification with Onion Protocol Implementation for Reliable Routing In Wireless Networks. *Int. J. Eng. Technol.* **2016**, *8*, 183–190.
22. De Cristofaro, E.; Ding, X.; Tsudik, G. Privacy-preserving querying in sensor networks. In Proceedings of the 2009 18th International Conference on Computer Communications and Networks, San Francisco, CA, USA, 3–6 August 2009; pp. 1–6.

23. Carbunar, B.; Yu, Y.; Shi, W.; Pearce, M.; Vasudevan, V. Query Privacy in Wireless Sensor Networks. *ACM Trans. Sen. Netw.* **2010**, *6*, 1–34. [CrossRef]
24. Feng, L.; Zhang, H.; Lou, L.; Chen, Y. A blockchain-based collocation storage architecture for data security process platform of WSN. In Proceedings of the 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD)), Nanjing, China, 9–11 May 2018; pp. 75–80.
25. Roman, R.; Alcaraz, C.; Lopez, J.; Sklavos, N. Key management systems for sensor networks in the context of the Internet of Things. *Comput. Electr. Eng.* **2011**, *37*, 147–159. [CrossRef]
26. Moinet, A.; Darties, B.; Baril, J.L. Blockchain based trust & authentication for decentralized sensor networks. *arXiv* **2017**, arXiv:1706.01730.
27. Casado-Vara, R.; de la Prieta, F.; Prieto, J.; Corchado, J.M. Blockchain framework for IoT data quality via edge computing. In Proceedings of the 1st Workshop on Blockchain-Enabled Networked Sensor Systems, Shenzhen China, 4 November 2018; pp. 19–24.
28. Islam, A.; Al Amin, A.; Shin, S.Y. FBI: A Federated Learning-Based Blockchain-Embedded Data Accumulation Scheme Using Drones for Internet of Things. *IEEE Wirel. Commun. Lett.* **2022**, 11, 972–976. [CrossRef]
29. Shahid, A.R.; Pissinou, N.; Staier, C.; Kwan, R. Sensor-chain: A lightweight scalable blockchain framework for internet of things. In Proceedings of the 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Atlanta, GA, USA, 14–17 July 2019; pp. 1154–1161.
30. HUANG, H.; Xiaoxiao, W.; Gangqiang, L. Anomaly detection and location of malicious node for IoT based on smart contract in blockchain network. *Chin. J. Internet Things* **2020**, *4*, 58–69.
31. Szabo, N. Formalizing and securing relationships on public networks. *First Monday* **1997**, *2* . [CrossRef]
32. Buterin, V. Ethereum white paper. *GitHub Repos.* **2013**, *1*, 22–23.
33. Ethereum Clique Proof-of-Authority Consensus Protocol. Available online: https://github.com/ethereum/EIPs/issues/225 (accessed on 25 May 2022).
34. Ethereum Light Client Protocol. Available online: https://eth.wiki/concepts/light-client-protocol (accessed on 25 May 2022).
35. Hrovatin, N.; Tošić, A.; Mrissa, M.; Vičič, J. A General Purpose Data and Query Privacy Preserving Protocol for Wireless Sensor Networks. *arXiv* **2021**, arXiv:2111.14994.
36. Merkle, R.C. A digital signature based on a conventional encryption function. In *Conference on the Theory and Application of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1987; pp. 369–378.
37. GAMS Indoor Air Quality Dataset. Available online: https://github.com/twairball/gams-dataset (accessed on 24 April 2022).
38. Ristanović, M.; Miloš, M. Experimental Protocol for Assessing the Relation Between Indoor Air Quality and Living Unit Conditions in AAL. *Teh. Vjesn.* **2022**, *29*, 536–541.
39. Wu, Z.; Ma, C.; Shi, X.; Wu, L.; Dong, Y.; Stojmenovic, M. Imputing missing indoor air quality data with inverse mapping generative adversarial network. *Build. Environ.* **2022**, *215*, 108896. [CrossRef]
40. Saini, J.; Dutta, M.; Marques, G. ADFIST: Adaptive Dynamic Fuzzy Inference System Tree Driven by Optimized Knowledge Base for Indoor Air Quality Assessment. *Sensors* **2022**, *22*, 1008. [CrossRef]
41. Witten, I.H.; Frank, E.; Hall, M.A.; Pal, C.J. *Online Appendix for "Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques"*, 4th ed.; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2016.
42. Quinlan, J.R. *Learning With Continuous Classes*; World Scientific: Singapore, 1992; pp. 343–348.
43. Wang, Y.; Witten, I.H. Inducing Model Trees for Continuous Classes. In *Poster Papers of the 9th European Conference on Machine Learning*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 128–137.
44. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
45. 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. [CrossRef]
46. 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; pp. 71–80.
47. Hoeffding, W. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*; Springer: Berlin/Heidelberg, Germany, 1994; pp. 409–426.
48. Montiel, J.; Read, J.; Bifet, A.; Abdessalem, T. Scikit-Multiflow: A Multi-output Streaming Framework. *J. Mach. Learn. Res.* **2018**, *19*, 1–5.
49. Halford, M.; Bolmier, G.; Sourty, R.; Vaysse, R.; Zouitine, A. Creme, a Python Library for Online Machine Learning. 2019. Available online: https://pypi.org/project/creme/ (accessed on 8 March 2022).
50. Montiel, J.; Halford, M.; Mastelini, S.M.; Bolmier, G.; Sourty, R.; Vaysse, R.; Zouitine, A.; Gomes, H.M.; Read, J.; Abdessalem, T.; et al. River: Machine Learning for Streaming Data in Python. 2021. Availableonline:https://www.jmlr.org/papers/volume22/20-1380/20-1380.pdf (accessed on 17 March 2022).
51. Ikonomovska, E.; Gama, J.; Džeroski, S. Learning model trees from evolving data streams. *Data Min. Knowl. Discov.* **2011**, *23*, 128–168. [CrossRef]
52. Hrovatin, N.; Tošić, A.; Vičič, J. PPWSim: Privacy preserving wireless sensor network simulator. *SoftwareX* **2022**, *18*, 101067. [CrossRef]

53. nsnam. ns-3, a Discrete-Event Network Simulator for Internet Systems—Version 3.32. 1 October 2021. Available online: https://www.nsnam.org/ (accessed on 11 February 2022).
54. Chernyshev, M.; Baig, Z.; Bello, O.; Zeadally, S. Internet of things (iot): Research, simulators, and testbeds. *IEEE Internet Things J.* **2017**, *5*, 1637–1647. [CrossRef]
55. Clausen, T.; Jacquet, P.; Adjih, C.; Laouiti, A.; Minet, P.; Muhlethaler, P.; Qayyum, A.; Viennot, L. Optimized Link State Routing Protocol (OLSR). Rfc, INRIA. 2003. Available online: https://hal.inria.fr/file/index/docid/471712/filename/5145.pdf (accessed on 23 April 2022).
56. Libsodium The Sodium Crypto Library. Available online: https://libsodium.gitbook.io/doc/ (accessed on 28 May 2021).
57. Alrubei, S.M.; Ball, E.A.; Rigelsford, J.M.; Willis, C.A. Latency and performance analyses of real-world wireless IoT-blockchain application. *IEEE Sens. J.* **2020**, *20*, 7372–7383. [CrossRef]
58. Bernstein, D.J. Curve25519: New Diffie-Hellman speed records. In *International Workshop on Public Key Cryptography*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 207–228. [CrossRef]
59. Ikonomovska, E.; Gama, J.; Zenko, B.; Dzeroski, S. Speeding-Up Hoeffding-Based Regression Trees with Options. ICML. 2011. Available online: https://openreview.net/forum?id=ByVpXjZ_WS (accessed on 17 April 2022).