# Distributed Data Validation Network in IoT: A Decentralized Validator Selection Model


Mohammed B. M. Kamel
Eotvos Lorand University
Budapest, Hungary
University of Applied Sciences Furtwangen
Furtwangen, Germany
mkamel@inf.elte.hu

Kevin Wallis
University of Freiburg
Freiburg, Germany
University of Applied Sciences Furtwangen
Furtwangen, Germany
kevin.wallis@hs-furtwangen.de

Peter Ligeti
Eotvos Lorand University
Budapest, Hungary
turul@cs.elte.hu

Christoph Reich
University of Applied Sciences Furtwangen
Furtwangen, Germany
christoph.reich@hs-furtwangen.de



## ABSTRACT

The generated real-time data on the Internet of Things (IoT) and the ability to gather and manipulate them are positively affecting various fields. One of the main concerns in IoT is how to provide trustworthy data. The data validation network ensures that the generated data by data sources in the IoT are trustworthy. However, the existing data validation network depends on a centralized entity for the selection of data validators. In this paper, a decentralized validator selection model is proposed. The proposed model creates multiple clusters using the distributed hash table (DHT) technique. The selection process of data validators from different clusters in the model is done randomly in a decentralized scheme. It provides a global method of assignment, selection, and verification of the selected validators in the network.



## CCS CONCEPTS

• **Networks** → **Network security**; **Peer-to-peer networks**.

## KEYWORDS

Data Validation, Data Validation Network, Cluster-Based Data ÂăValidation, Distributed Hash Table, Internet of Things, Industrial Internet of Things, Big Data

**ACM Reference Format:**
Mohammed B. M. Kamel, Kevin Wallis, Peter Ligeti, and Christoph Reich. 2020. Distributed Data Validation Network in IoT: A Decentralized Validator Selection Model. In *IoT2020: 10th International Conference on the Internet of Things, October 06–09, 2020, Malmo, Sweden.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3410992.3411027




*IoT 2020, October 06–09, 2020, Malmo, Sweden*



ACM ISBN 978-1-4503-8758-3/20/10…$15.00
https://doi.org/10.1145/3410992.3411027


## 1 INTRODUCTION

The generated real-time data on the Internet of Things (IoT) and the ability to gather and manipulate them are positively affecting various fields. The generated data by resources (i.e. a data source) in IoT have to be validated to provide trustworthy data. The data can be validated on the data source itself (when powerful enough), the gateways, or in separate data validator nodes. There are a few reasons to create a separate data validation network. Firstly, not all generated data by resources will be needed by clients. Therefore, processing only consumed data by clients considered more efficient than processing all generated data. Besides, hundreds of resources can be connected to a gateway, and processing all incoming data might add significant overhead. Second, handling the generated data locally by a gateway causes the loss of generality as each data will be processed locally without taking into consideration other generated data that is processed by other gateways. Furthermore, there is a lot of unused computational power e.g. machines, production pcs, tablets at a shop floor, etc. that can be used for data validation.

To avoid dependency on a single validation component and thus prevent the single point of failure and attack, several distributed validator nodes are used. Distributed data validation network uses existing processing capacities, e.g. unused computers in production environments or rarely used servers for the validation of data. The client accepts the data that is received from a data source after being validated by several data validators in the distributed data validation network. A data validator registers its services in a centralized entity, and clients select the data validators based on their registered services in the server. The server as a trusted centralized entity might turn into a bottleneck in the system. Therefore, in this paper we address the following research question: *How can the data validators in the distributed data validation network be selected randomly without involving a trusted third party?*

To answer this research question, a decentralized validator selection model is proposed. The proposed model utilized the Distributed Hash Table (DHT) to create multiple overlays (i.e. clusters). The data validators reside in the overlays based on their generated identifiers and clients select several data validators randomly without any centralized entity. The rest of this paper is organized as follows. The next section summarizes the efforts in the current research field of the distributed data validation network. Section 3 explains the

preliminaries. In section 4 the proposed model of decentralized data validator selection is described. Section 5 discusses the performance analysis of the identifier assignment in the model. Finally, Section 6 presents our conclusions.

## 2 RELATED WORK

Gould et al. [9] patented a data profiling system that reads data from a data source, computes a data characterization summary, and stores the data based on the characterization information. On one hand, they do not consider malicious manipulation of data and on the other hand, no distribution of the profiling module is considered. A survey of different data profiling techniques is given by Abedjan et al. [1]. Their work uses the same definition as Johnson: *"Data profiling refers to the activity of creating small but informative summaries of a database"* [10]. In comparison, we do not only consider data from databases in data profiling, but also data that is validated in real-time by validator nodes.

Thottan et al. [20] and Hood et al. [8] use a distributed agents approach to detect anomalies in the network proactive. The distributed agents approach was able to detect a file server failure 12 minutes before it occurred. Despite the use of a distributed agents approach, no consideration of corrupted agents is made in comparison to the data validation network.

A comparison between the distributed data validation network and the blockchain technology is given in [21]. The main difference is that the client can decide on its own which information to store. On the one hand, it knows all connected nodes and on the other hand, it receives all data validation results. Thus, a blockchain can be used to support the client or the whole distributed data validation network but is not required. Furthermore, the use of a simple database is sufficient for logging functionality. Authors in [13] utilized blockchain and DHT to create a distributed open charging system of electric vehicles. Blockchain is used to store the timestamps while DHT is used to store the actual system data such as the status of charge points and electric vehicles. The paper focuses on specific subset of data validation network, the data validity through validation rules, and proposes the distributed data validation in DHT to define, publish, agree and enforce validation rules.

In this paper, a novel adoption of DHT in a distributed data validation network is discussed that focuses on the selection of the data validator nodes in the distributed data validation network. The assumption in the distributed data validation network is that some of the data validators might be corrupted. Therefore, one of the main concerns of the data validation network is how to select several random data validators. This can be done by a trusted selector server. Although the selector server helps to select some random validators from the data validation network, this centralized point might turn into a bottleneck as well as a single point of failure and attack. Peer-to-peer schemes can replace the centralized entity and distribute the process among all nodes. Protocols such as Chord [19] and Kademila [15] use the DHT technique to create the peer-to-peer network. DHT based systems assign a seemingly unique identifier to each node that joins the network. As a result, each data validator in the data validation network has a unique identifier and the clients select randomly some validators based on their identifiers. The challenge in the peer-to-peer scheme is how to generate and assign valid identifiers to the nodes in the network. A possible solution is the use of a centralized entity [3] to generate and assign the certified identifiers. The centralized identity assignment entity has the same drawback of being the bottleneck and single point of failure and attack. As a replacement to the trusted centralized entity, the identifier generation can be done locally by each node in the network. The proposed model allows the data validators to generate their identifiers, and the clients to select the data validators randomly based on their identifiers and validate the selected identifiers.

## 3 PRELIMINARIES

The following section explains the two essential technologies: the distributed data validation network and the distributed hash table. These technologies are the basis for the following chapters.

### 3.1 Distributed Data Validation Network

Data quality is an essential prerequisite for data analysis. To ensure data quality, there are several quality dimensions such as completeness, uniqueness, timeliness, validity, accuracy, and consistency. [4][17] The most common causes of poor data quality are system changes, software errors, erroneous data as well as data integration and data migration.[6][16] Intentional manipulation of the data is often not listed among the causes. This is where the distributed data validation network [21] is applied. It uses existing processing capacities, e.g. unused computers in production environments or rarely used servers for the validation of data.

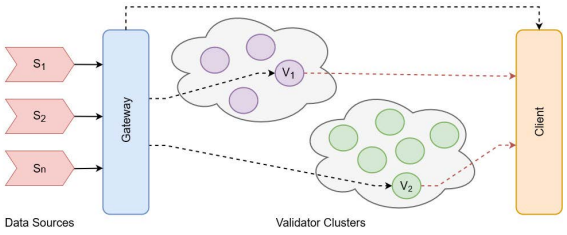


**Figure 1: Distributed Data Validation Network Structure**

To avoid dependency on a single validation component and thus prevent the single point of failure, several distributed validator nodes are used. Figure 1 shows the structure of the Distributed Data Validation Network. On the left side the data sources $s_1, s_2, ..., s_n$ (e.g. sensors of machines) are located which are connected to gateways. There can be several gateways with different data sources. The communication with the validation network is always performed via a gateway because sensors have no network capability on the one hand and the other hand, not enough computing power to encrypt the communication. Besides, the gateways are classified as trustworthy, and together with the clients on the right side, they form the trust anchors. The validators $v_1, v_2, ..., v_m$ are in the middle in two different clusters. Either rule-based approaches or machine learning approaches can be used for validation. The client collects the validation results of the used validators and calculates a data quality value from them, which indicates whether the data should

be used or not. The data quality value can be classified as 0 (normal) and 1 (anomalous). The clusters in the data validation network correspond to a logical classification of the validators (e.g. by the operating system, location, etc.) and each validator is contained in one or more clusters.

To prevent an attacker from sending multiple false validation results to a client, a challenge-response method is used. The client randomly decides which validators from which clusters will provide the validation results for upcoming data quality considerations. Results from non-permitted validators can thus be dropped. The validators are selected according to their registered validation services on the Consul[1] server. The Consul server is a centralized entity, which is replaced by a decentralized approach in this paper.

The data validation network generates additional communication in the system. To keep the amount of data sent as low as possible, the actual data is sent directly from the gateway to the client and the selected validators. The selected validators therefore only need to forward their respective validation results to the client.

## 3.2 Distributed Hash Table

DHT is a distributed system that creates a structured peer-to-peer scheme in a network. The participating nodes in the network can join and leave the DHT at any specific time. Upon joining a new node, a new identifier is assigned to it. Using identifiers instead of other types of addressing (e.g. IPs) helps to balance the data storage among participating nodes without any centralized entity. In addition to load balancing, it solves the scalability by providing the service of generating the identifiers by the participating nodes themselves. There are several protocols to implement DHT such as Chord [19], Kademila [15], Pastry [18], and Tapestry [22].

DHT uses a large address space of integer numbers. The size of the address space depends on the fixed output size of the function that is used to generate the identifier. To achieve the random function of identifier generation and uniform distribution of data among all participating nodes a collision-resistant one-way hash function is used in DHT.

*Definition 3.1.* A function H(.) that maps an arbitrary length input $M$ into a fixed-length digest $d$ is called collision-resistant one-way hash function it satisfies the following properties:

- Given $M$, it is easy to compute $H(M)$.
- Given $d$, it is hard to find any $M$ s.t. $d = H(M)$.
- Given $d = H(M)$ and $M$, it is hard to find $M'$ s.t. $M' \neq M$ and $H(M) = H(M')$.
- It is hard to find two distinct messages $M'$ and $M''$ s.t. $M' \neq M''$ and $H(M') = H(M'')$.

If the solution can be computed in the polynomial time, therefore it is considered *easy* to compute. On the other hand, if there is no solution known to solve the problem in polynomial time, it is considered *hard* [5].

Similar to hash tables [14], the data in DHT is stored in key/value pairs. The value parameter includes any stored information about the data (e.g. the address of the data) and can be retrieved from the DHT based on its associated key. The key parameter of the key/value pair is generated by feeding specific information (e.g. the attribute of the stored data such as its name, its type, etc.) to the collision-resistant one-way hash function which produces a uniformly distributed randomized hash value that is used to determine the responsible node in the network of storing this specific pair. DHT has two implementation interfaces: put and get. The Put interface takes the key/value pair and stores this pair in the DHT. The Get interface takes a single parameter key and lookup in the DHT to retrieve the identifier of a node that is responsible to store the corresponding value to the given key. In the DHT the store (i.e. put interface) and lookup (i.e. get interface) operations are done with an upper bound of $O(log(N))$, in which $N$ is the number of nodes in the DHT. This feature guarantees that any participating node in DHT can store a pair of key/value or lookup based on a given key by routing through of maximum $log(N)$ nodes.

[1] https://www.consul.io/

# 4 PROPOSED DATA VALIDATION MODEL

## 4.1 Model Participants

There is a set of resources $\mathcal{S}$ that are considered the sources of data in the system. Each subset of resources in $\mathcal{S}$ is connected to the network through a gateway $w \in \mathcal{W}$. The sets of $\mathcal{S}$ and $\mathcal{W}$ are disjoint sets. At any given time the relation between members of $\mathcal{S}$ and members of $\mathcal{W}$ is many-to-one. Each $s \in \mathcal{S}$ generates the data and sends it through its directly connected gateway $w$. The data that is generated by members of $\mathcal{S}$ is consumed by the set of clients $C$. Each client $c \in C$ might ask for data that is generated by any non empty subset of $\mathcal{S}$. Some nodes are mostly idly grouped in a set $\mathcal{V}$. These nodes are used to perform the calculation required to validate the incoming data from any $s \in \mathcal{S}$. A client after discovering a resource (i.e. a data source), selects randomly several validators from $\mathcal{V}$ to perform the validation process of the data. The validation process is done by checking the data based on a set of thresholds or using machine learning techniques. Figure 2 shows the overall structure of the model.

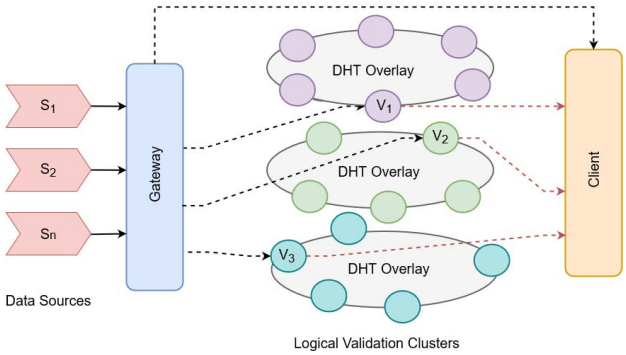


**Figure 2: DHT Overlays in Data Validation Network**

## 4.2 System Workflow

The IoT consists of a large number of resources that represent the sources of data in the network. Since these resources are distributed in different physical and logical parts, having a centralized entity to control all these resources is not feasible. In decentralized resource discovery [11, 12] the resources are registered in different parts

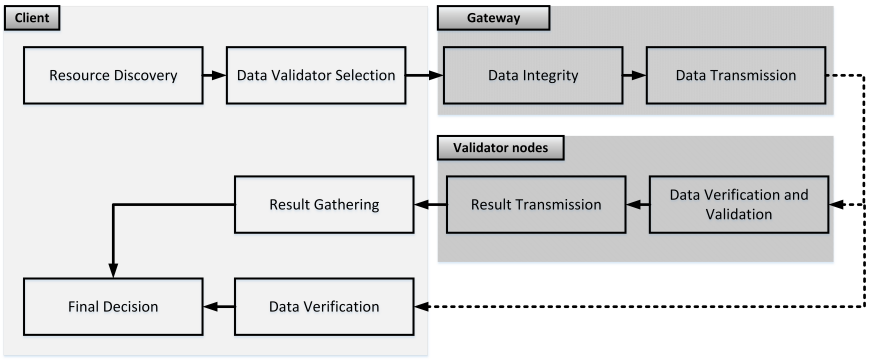


**Figure 3: System Workflow**

of the network and there is no trusted third party that control all registered resources. During the lookup phase, certain resources that provide specific data are discovered.

The data validation is done by a number of distributed nodes that represents the distributed data validation network. We assume that any validator node can validate any data. The client and after discovering the required resources selects randomly some data validators from different available clusters (i.e. DHT overlays) and requests the gateways to transmit the generated data by those resources directly to the client as well as to the selected data validators. A list of the selected data validators is passed to the corresponding gateways. The selection process of the data validators can be reissued after a specific period. In the next phase, the gateways digitally sign the data and transmit that to the requested client and the list of the selected data validators by the client. The sets gateways and clients are classified as trustworthy and together form the trust anchors.

The nodes in the distributed data validation network reside in different logical clusters using different DHT overlays. If selected by a client, each of the data validators in the distributed data validation network validates the data transmitted from resources through their connected gateways based on a set of validation rules. If the data rely upon the permitted range then the data considered valid, otherwise the data validation fails. The incoming data to the selected data validators are verified using the public key of the respective gateways. Then, the verified data are checked by the data validators to ensure that they meet expected standards and are within anticipated tolerances, using a configurable series of rules and properties.

The data that is received by a client is verified using the public key of the gateways. The verified data is considered valid depending on the decision of the selected data validators. Figure 3 illustrates the workflow of the proposed model.

## 4.3 Security Properties

Before defining the security properties we need to define assumptions regarding the participants of the model. The members of $\mathcal{W}$ and $C$ are considered to be *semi-honest*. The semi-honest entities are assumed to follow the protocol properly, but they are allowed to store the received data locally in an attempt to get more information from the stored data. The members of $\mathcal{W}$ receive the data from resources that are connected to them. These data will be passed to members of $C$ and $\mathcal{V}$ based on their request. The members of $C$ are the request initiators. Regarding these requests we need to assume that from the viewpoint of any node $c \in C$, the proportion of the resources in $\mathcal{S}$ and data validators in $\mathcal{V}$ can be assumed to be *malicious*. Some publicly verifiable data related to the validators can be *unforgeable*, in the sense that any $v' \in \mathcal{V} \setminus \{v\}$ can forge the related data $data_v$ of a given validator $v \in \mathcal{V}$ with negligible probability only. A function has a negligible success probability if it occurs with a probability smaller than any polynomial fraction [2]. The proposed model is assumed to achieve computational security, i.e. every probabilistic polynomial-time (PPT) adversary can break the security properties with negligible probability only.

The proposed model creates several clusters and includes the validators in the clusters based on their attributes. The goal for our proposed model is to allow any $c \in C$ to be able to select some data validators to be responsible for the validation of the data in a computationally non-predictable way. This is done through a novel multi-cluster model with random nodes for each validation process. Note that within this paper we don't suggest any countermeasures against fake data providing, rather we are focusing on the correct, unique, and verifiable assignment of identifiers of the validators which leads to random selection of data validators. The identifier assignment in the proposed model has to satisfy the following security properties:

- **Correctness**: The valid identifiers generated by semi-honest data validators can be verified by all clients.
- **Soundness**: The PPT validators can generate more than one verifiable identifiers for a given cluster with negligible probability only.

## 4.4 Model Description

The nodes in $\mathcal{V}$ have to be able to generate their identifiers to be used in the clusters of the network (i.e. a partial identifier for each cluster) without any centralized entity. Besides, the clients in $C$ have to be able to verify the identifier of any selected data validator $v \in \mathcal{V}$ at any given time. The requirements of the identifiers are as follows:

- Each validator node in the data validation network has to be able to generate only one valid identifier per cluster.

- A client has to be able to verify the identifier of a selected validator node without any centralized entity.

*Clusters.* There are $L$ clusters in the data validation network. Each cluster indicates a separate DHT overlay. The number of the clusters (i.e. $L$) is defined based on the use cases and available categories in a system. Each node in the data validation network has a set of attributes that allow it to be part of one or more available clusters in the data validation network. Therefore as illustrated in Figure 4 a physical node might be part of one or more clusters (i.e. DHT overlays) based on its attributes. The attributes of a node $v \in \mathcal{V}$ are defined by an *Attribute Vector (AV)*. The $AV$ of a validator node indicates the clusters that can be joined based on its attributes. $AV$ is public and can be checked and verified by any node in the system.

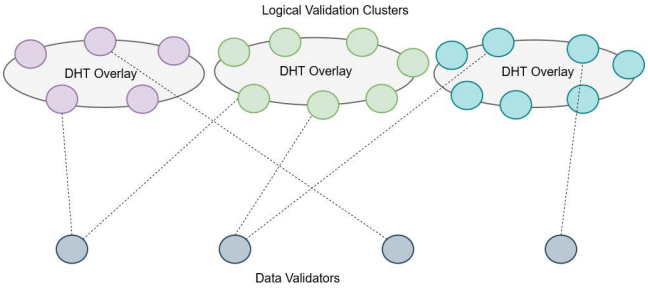


**Figure 4: Physical and Logical Data Validators**

The model uses a collision-resistant one-way hash functions $H(.)$ (Definition 3.1) to generate the identifiers of the members of $\mathcal{V}$. Each node can have up to $2^m$ partial identifiers derived from its unique identity (e.g. its IP). The $m$ parameter defines the fixed number of bits required to index all clusters in the data validation network and chooses in a way such that $L \leq 2^m$. For example, in an organization that has eight clusters in the system, the value of $m$ parameter will be set to three. A data validator node in this network might need to generate eight partial identifiers. The partial identifiers of a node are used to reside it in the different clusters in the data validation network. The clusters are categorized based on a distinct feature, like types of the operating systems or the locations of a data validator. The most important issue here is that a category has to be chosen in a way that there is a possibility to verify that a data validator has legitimately joined this specific overlay, i.e. its current clusters match its attribute vector. Considering an organization with eight clusters, table 1 shows the details of used overlays in the system.

*Identifier Assignment Method (IDAM).* During the design of the Identifier Assignment Method *(IDAM)*, two main cases for identifier generation have been studied: either each data validator has the same identifier in all different clusters it belongs to, or it has different identifiers (called partial identifiers) for each cluster it belongs to. In the first case that a data validator has the same identifiers, the clients to select $t$ different number of physical data validators from $t$ clusters have to generate $t$ random identifier. In the second case that a data validator has $t$ different partial identifiers in the clusters it belongs to, the client to select $t$ different number of data validators from $t$ clusters has to generate one random identifier. Since the ratio of clients selecting a set of data validators is higher than the ratio of data validators' life cycle, then adopting the partial identifiers as in the second case is more efficient. Upon joining a new data validator node $v^*$ and depending on its attribute vector $AV_{v^*}$, the node $v^*$ can be part of one or more clusters in the data validation network. The *(IDAM)* is used to derive partial identifiers of $v^*$ using hash function $H(.)$. Different collision-resistant one-way hash functions such as SHA-256 or SHA-512 [7] can be used in *IDAM* during the process of generating the partial identifiers of data validators in the different clusters. There are two versions of *IDAM*, namely *IDAMv1* and *IDAMv2*. The main differences in the two methods are the usage of a hash function to generate the partial identifiers that distinguish them from each other as follows:

**Table 1: Example of an organization with $L = 8$ and $m = 3$**

| Cluster | Type | Category | AV value |
|---|---|---|---|
| 0 | OS | Windows | 1xxxxxxx |
| 1 | OS | Linux | x1xxxxxx |
| 2 | OS | Mac | xx1xxxxx |
| 3 | OS | Others | xxx1xxxx |
| 4 | Location | location 1 | xxxx1xxx |
| 5 | Location | location 2 | xxxxx1xx |
| 6 | Location | location 3 | xxxxxx1x |
| 7 | Location | location 4 | xxxxxxx1 |

- *IDAMv1*: Takes the address and the attribute vector as an input and uses the hash function $H(.)$ once to derive $L$ different partial identifiers by applying different permutations on the hash value.
- *IDAMv2*: Takes the address, the attribute vector, and the cluster number as an input and uses the hash function $L$ times to generate $L$ different partial identifiers.

The *IDAMv1* (see Figure 5) takes the address and the attribute vector of $v^*$ as input and using the hash function $H(.)$ generates its hash value. Then permutes the output of the hash function using $Perm_x(.)$, which is a fixed and public permutation for the cluster $x$. The binary value of the cluster number is then prepended to the output of the permutation to form the partial identifier of data validator $v^*$ in cluster $x$:

$$idx_{v^*} = IDAMv1(v^*, x) = (x|Perm_x(H(add_{v^*}|AV_{v^*}))) \quad (1)$$

The *IDAMv2* (see Figure 6) takes the address and the attribute vector of $v^*$ along with the cluster number as input and using the hash function $H(.)$ generates its hash value. The resulted hash value will be the partial identifier of $v^*$ in the corresponding cluster:

$$idx_{v^*} = IDAMv2(v^*, x) = (x|H(add_{v^*}|AV_{v^*}|x))) \quad (2)$$

*Joining the data validation network.* When a new node $v^*$ wants to join the network, it first chooses the clusters that are legitimate to be part of. As the clusters are designed in non-overlapped categories, it is easy for a node to choose its proper clusters that can join based on its attribute vector $AV_{v^*}$. For instance, if a node in the example organization given on table 1 that runs a Linux operating system and resides in location 1 joins the data validation network (i.e.

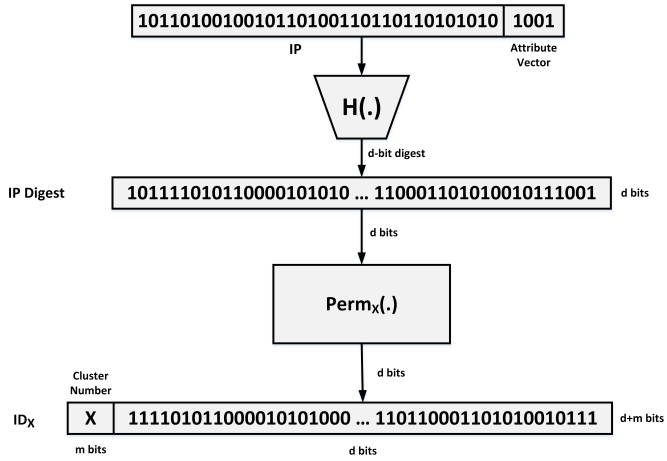


**Figure 5: Generation process of partial identifiers in IDAMv1**

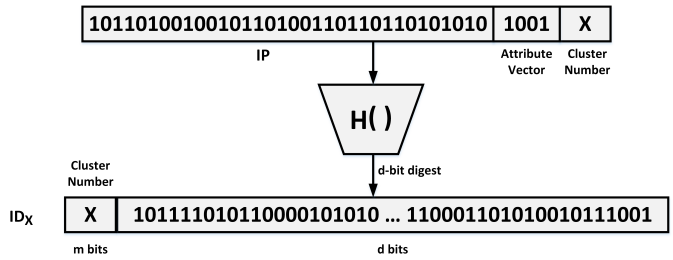


**Figure 6: Generation process of partial identifier of Cluster in IDAMv2**

$AV_{v^*} = 01001000$), it can be part of cluster 1 and 4. It will generate its partial identifiers based on its address $add_{v^*}$ and using *IDAM* to join cluster 1 (using generated $id1_{v^*}$) and cluster 4 (using generated $id4_{v^*}$). The new node can join these clusters using generated partial identifiers by creating two pairs as $\langle id1_{v^*}, add_{v^*} \rangle$ and $\langle id4_{v^*}, add_{v^*} \rangle$ to be added to the respective clusters. After that, the validator $v^*$ publishes its partial identifiers $(v^*, id1_{v^*}, id4_{v^*})$.

*Registering in the overlays.* A node after joining the network and generating its relevant identifiers has to register in the overlays. The registration is done by issuing a lookup process (*get_validator* procedure) for its partial identifiers in the overlays that it belongs to. Therefore, the $v^*$ runs the *get_validator* ($id1_{v^*}$) and *get_validator* ($id4_{v^*}$) in cluster 1 and cluster 4, respectively. Since $v^*$ is the only node that currently knows the result, the request after passing through a few intermediate nodes will finally reach the $v^*$. The $v^*$ returns the pairs $\langle id1_{v^*}, add_{v^*} \rangle$ and $\langle id4_{v^*}, add_{v^*} \rangle$ in cluster 1 and cluster 4, respectively. As a result, the $v^*$ information will be part of the respective clusters stored in multiple nodes in the network.

*get_validator interface.* Each node in a data validation overlay stores a pointer to its close and distant nodes in $d$ lists of size $k$ each. The $d$ is equal to the number of bits in the output digest of the used hash function *H(.)* and $k$ parameter is chosen such that the members in any given subset $M \subset \mathcal{V}$ with cardinality $k$ is unlikely to fail (i.e. being inaccessible or offline). The $j$th list includes the pointers to maximum $k$ nodes that the $j-1$ prefix bits of their identifiers matches, where $j \in [1, d]$. XOR metric is used for the distance calculation. The identifiers are XORed and the result indicates the distance between any two identifiers in the network. This means that the closest node to node $n$ is itself, $n \oplus n = 0$, and the distance between any two nodes are symmetric, $x \oplus y = y \oplus x = z$

The *get_validator* interface takes an identifier as input and returns the communication information (i.e. IP and port) and the identifier of the closest node to the given identifier. Similar to *find_node* procedure in Kademlia[15], a receiver of the *get_validator* procedure either returns the tuple of (IP, port, and identifier) of the node that its identifier matches the given parameter or the $\alpha$ closest nodes that it knows to the required identifier. $\alpha$ is a system-wide concurrency parameter with a default value set to 3. The *get_validator* procedure continues to request the $\alpha$ closest available nodes in the local lists until retrieving the closet node to the requested identifier. DHT guarantees that the *get_validator* procedure returns the answer within $O(log(N))$ steps, in a network with $N$ nodes.

*Select the data validators.* . Data validators are selected randomly. During the selection process, one or more different validators can be selected from each cluster. A client $c \in C$ generates a random identifier $r$. Then the client $c$ lookup in the system to find the closest data validator to $r$ in each cluster. The data validators are selected and verified by the clients as follows:

(1) The client $c$ chooses a random identifier, $r$.
(2) Then $c$ lookup in the different clusters (i.e. DHT overlays) to get the communication information (IP and port) and the identifier of the data validator that its identifier in the overlay matches or closest to $r$ and retrieved it from the overlay.
(3) The client verifies the retrieved data (communication address of the data validator) by verifying its *attribute vector* and recomputing its identifier (and its particular partial identifier) using the corresponding *IDAM*.
(4) After verification, the client sends the list with the communication information to the gateway and the gateway uses this information to forward the signed data to the correct validators. Furthermore, the gateway also sends validators information about the target client.

## 4.5 Advantages of the Proposed Model

Using a distributed data validation network compared to a normal connection without data validation capability offers the following advantages:

- *Data Quality Improvement:* Validation of the data for anomalies and thus an increased quality of the data for further processing, e.g. predictive maintenance or condition monitoring.
- *Single Point of Failure Prevention:* By using a distributed data validation approach, the dependency on a single validation node is split, resulting in a lower success probability for an attack on the system.
- *DoS Prevention:* By using clusters based on validation node properties and random selection among those validators, a

DoS attack on specific validator nodes (i.e. physical validators) or a specific property (e.g. on Windows systems) cannot shut down the entire validation system.
- *Confidentiality:* Encrypted communication and thus more difficult information theft.
- *Integrity:* The transmitted sensor data are digitally signed and can therefore neither be manipulated nor enriched with additional data.
- *Availability:* Due to better data quality, predictions become more accurate and systems are better utilized. This increases the overall availability of all systems based on a distributed data validation system.
- *Non-Repudiation:* By using a unique ID assignment mechanism, (e.g. such as Consul with certificates), it is ensured that the individual validation nodes of the validation network cannot deny their respective validation results.
- *Authenticity:* In addition to non-repudiation, the unique ID assignment mechanism with non-repudiation property also guarantees the identity and thus the authenticity of the validation nodes.
- *Clusters:* By using clusters based on the attributes of validator nodes, the probability of success for an attack is reduced. An attack uses a vulnerability in a specific system, for example, a vulnerability in Windows. Selecting validator nodes from different clusters (for example, Windows and Linux) ensures that an attacker will not succeed based on a single known vulnerability.

One of the core parts of a data validation network is the validator selection phase. Distributed data validation network such as [21] uses a central validation node identifier generation and selection approach. Although this centralized approach can be partially distributed and protected using redundant servers (e.g. mirrored Consul servers), the distributed approach we show here offers some advantages:

- *Independence from a Central Instance:* The use of a central instance is no longer necessary and thus the single point of failure in identifier generation and selection disappears.
- *Different Identifiers per Cluster:* When using clusters, an individual identifier per cluster can be generated quickly for a validation node. This results in a performance gain as long as the number of identifier renewals (e.g. daily) is less than the selection of identifiers in the same period.
- *Verification of a Node's Identity:* The generated identity of a validation node can be verified by any other node (see Section 5.1).

Despite the advantages mentioned, there are still some research questions that need further clarification. These include *a*) the use of validation nodes with different validation capabilities and *b*) the verification of the correctness of the attribute vector.

# 5 EVALUATION

For the evaluation, a proof for the correctness and a proof for the soundness are given. Furthermore, a timing analysis that compares IDAMv1 with IDAMv2 and shows their respective timing benefits is done.

## 5.1 Security Analysis

In the following, the proofs for correctness and soundness are given.

Theorem 5.1. *The system satisfies correctness.*

Proof. Suppose that a *semi-honest* node $v$ joins the data validation network, with an address $add_v$ and the attribute vector $AV_v$. Assume that $v$ generates and publishes its partial identifier $idx_v$ for cluster $x$ by $IDAMv1$ or $IDAMv2$ and let $c \in C$ be any client. Since the functions $H(.), Perm_x(.)$ and the data $add_v, AV_v, idx_v$ are publicly known, the client $c$ can verify whether $id_x$ is generated with $IDAMv1$ or $IDAMv2$, respectively. This completes the proof. □

Theorem 5.2. *If the address and the attribute vectors of every validator are unforgeable, $H(.)$ is a collision-resistant one-way hash function and for different clusters $x \neq y$ we have $Perm_x \neq Perm_y$, then the system satisfies soundness.*

Proof. Let $v \in \mathcal{V}$ be any validator and let $\mathcal{X}$ be the set of clusters $v$ is able to join. Every semi-honest $v$ can generate one verifiable identifier $idx_v$ for every $x \in \mathcal{X}$ as a consequence of Theorem 5.1 and semi-honest $v$ will generate only one identifier for every cluster. Hence we can assume that $v$ is malicious. The generation of one correct identifier per cluster is also trivially possible for malicious validators as well. Now suppose that the $v$ is able to generate a second identifier $idx'_v \neq idx_v$ for some cluster $x \in \mathcal{X}$. First, suppose that $v$ uses $IDAMv1$. Then since $Perm_x \neq Perm_y$ for $y \in \mathcal{X} \setminus \{x\}$ from 2 we get that $v$ can find $add'_v$ and $AV'_v$ with $idx'_v = (x|H(add'_v|AV'_v))$. This means that $v$ is either able to find a collision in $H(.)$ or can forge its address and attribute vector which is non-negligible probability. Second, suppose that $v$ uses $IDAMv2$. Then $v$ can find $add''_v$ and $AV''_v$ with $idx'_v = H(add''_v|AV'''_v|x))$. Similarly as above, then $v$ is either able to find a collision in $H(.)$ or can forge its address and attribute vector which is non-negligible probability. □

## 5.2 IDAM Analysis

For the practical consideration of $IDAMv1$ and $IDAMv2$, a prototypical implementation of the two approaches was developed. The implementation is based on `C` and uses the hash function of the OpenSSL[2] (OpenSSL 1.1.1d) library. The hash function is SHA-256 and the input is an IP address (four `unsigned chars` - corresponding to 32 bits, see Listing 1).

**Listing 1: IP Address Datatype**

```
struct IP {
    unsigned char a;
    unsigned char b;
    unsigned char c;
    unsigned char d;
};
```

The tests are performed on a Raspberry PI 3 Model B Vi. 2. A unique identifier is calculated for 10,000 different IP addresses. Assuming that the permutation function $Perm_x(.)$ of $IDAMv1$ is shifting $x$ times, The results of the identifier generations are listed

[2]https://www.openssl.org/

in Table 2. Line one $Default$ is the time for calculating a hash value without additional computational steps. It is shown that the calculation speed, even on a less powerful processing unit, is very fast and below 10 ms. Therefore, both approaches $IDAMv1$ and $IDAMv2$ are sufficient for real-world applications.

**Table 2: Simulation results for $IDAMv1$ and $IDAMv2$**

| Method | Summarized Duration | Averaged Duration |
|---|---|---|
| $Default$ | 40.994 ms | 0.004 ms |
| $IDAMv1$ | 76.405 ms | 0.008 ms |
| $IDAMv2$ | 60.639 ms | 0.006 ms |

In the given table, a separate identifier was created for each cluster. The advantage of the $IDAMv1$ approach is that as soon as an identifier has to be prepared for several clusters, only a single hashing process is required. All additional identifiers are generated from the existing identifier using the permutation function $Perm_x(.)$ in each cluster. Assuming that $n$ cluster identifiers are required, a hashing $h$ takes 0.004ms, an average shifting $s$ takes 0.004ms (0.008ms - 0.004ms = 0.004ms) and an $IDAMv2$ process $z$ takes 0.006 ms, the following applies:

$$n * z > h + n * s \tag{3}$$

$$n > \frac{h}{z - s} \tag{4}$$

Thus, if $n > \frac{h}{z-s}$, $IDAMv1$ is faster than $IDAMv2$, which in the present measurements mean that as soon as $n \geqslant 3$ $IDAMv1$ is faster.

## 6 CONCLUSION

In this paper, a decentralized model for data validator selection using DHT technology in distributed data validation network has been proposed. The proposed model creates a set of clusters in the data validation network that are created by using multiple DHT overlays. Each data validator and depending on its attribute vector can be part of one or more clusters. The proposed model allows data validators to generate their identifiers without any centralized entity. These identifiers can be later verified by the clients. There are two distributed identifier generation methods in the model, namely IDAMv1 and IDAMv2. The analysis showed that the IDAMv1 is faster than IDAMv2 if there are more than two clusters. Using a randomly generated identifier, the clients can select randomly several data validators from different clusters. The selected data validators are then used to validate the generated and transmitted data by IoT resources.

Although the distributed model can be a replacement for the centralized approach, some other issues have to be studied in future works. Among them are the selection and creation of the clusters, verification of the attribute vector of each data validator, and secure transmission of the decisions between the selected data validators and the clients.

## ACKNOWLEDGMENTS

This research has been partially supported by project no. ED_18-1-2019-0030 (Application-specific highly reliable IT solutions) has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme funding scheme and by the Ministry of Science, Research and the Arts Baden-Württemberg Germany.

## REFERENCES

[1] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. 2015. Profiling relational data: a survey. *The VLDB Journal* 24, 4 (2015), 557–581.
[2] Nirdosh Bhatnagar. 2019. *Mathematical Principles of the Internet, Two Volume Set.* CRC Press.
[3] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S Wallach. 2002. Secure routing for structured peer-to-peer overlay networks. *ACM SIGOPS Operating Systems Review* 36, SI (2002), 299–314.
[4] DAMA UK Working Group. 2013. *The Six Primary Dimensions for Data Quality Assessment.* Technical Report. https://www.whitepapers.em360tech.com/wp-content/files_mf/1407250286DAMAUKDQDimensionsWhitePaperR37.pdf
[5] Ivan Bjerre Damgård. 1989. A design principle for hash functions. In *Conference on the Theory and Application of Cryptology*. Springer, 416–427.
[6] Leah Davidson. 2019. *What Is Data Quality and Why Does it Matter?* https://www.springboard.com/blog/data-quality/
[7] Don Eastlake and Tony Hansen. 2006. US secure hash algorithms (SHA and HMAC-SHA).
[8] Hood, Cynthia S and Ji, Chuanyi. 1998. Intelligent agents for proactive fault detection. *IEEE Internet Computing* 2, 2 (1998), 65–72.
[9] Paul Bay Joel Gould, Carl Feynman. U.S. Patent 8868580B2, Oct. 2014. Data Profiling.
[10] Theodore Johnson. 2009. *Data Profiling.* Springer US, Boston, MA, 604–608. https://doi.org/10.1007/978-0-387-39940-9_601
[11] Mohammed B. M. Kamel, Bruno Crispo, and Peter Ligeti. 2019. A Decentralized and Scalable Model for Resource Discovery in IoT Network. In *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 1–4.
[12] Mohammed B. M. Kamel, Peter Ligeti, and Christoph Reich. 2020. Private/Public Resource Discovery for IoT: A Two-Layer Decentralized Model. In *The 12th Conference of PhD Students in Computer Science*. SZTE.
[13] Benedikt Kirpes, Micha Roon, and Christopher Burgahn. 2019. Distributed data validation for a key-value store in a decentralized electric vehicle charging network. (2019).
[14] Ward Douglas Maurer and Theodore Gyle Lewis. 1975. Hash table methods. *ACM Computing Surveys (CSUR)* 7, 1 (1975), 5–19.
[15] Petar Maymounkov and David Mazieres. 2002. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*. Springer, 53–65.
[16] Irene Mikhailouskaya. 2020. *Your Guide to Data Quality Management.* https://www.scnsoft.com/blog/guide-to-data-quality-management
[17] Leo L. Pipino, Yang W. Lee, and Richard Y. Wang. 2002. Data Quality Assessment. *Commun. ACM* 45, 4 (April 2002), 211âĂŞ218. https://doi.org/10.1145/505248.506010
[18] Antony Rowstron and Peter Druschel. 2001. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer, 329–350.
[19] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. 2001. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review* 31, 4 (2001), 149–160.
[20] Thottan, Marina and Ji, Chuanyi. 1998. Proactive anomaly detection using distributed intelligent agents. *Ieee network* 12, 5 (1998), 21–27.
[21] K. Wallis, F. Schillinger, C. Reich, and C. Schindelhauer. 2019. Safeguarding Data Integrity by Cluster-Based Data Validation Network. In *2019 Third World Conference on Smart Trends in Systems Security and Sustainablity (WorldS4)*. 78–86.
[22] Ben Y Zhao, Ling Huang, Jeremy Stribling, Sean C Rhea, Anthony D Joseph, and John D Kubiatowicz. 2004. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on selected areas in communications* 22, 1 (2004), 41–53.