computers

MDPI

Article

Trustworthiness of Dynamic Moving Sensors for Secure Mobile Edge Computing

John Yoon

Math/Computer Science Department, Mercy College, New York, NY 10522, USA; jyoon@mercy.edu

Received: 17 September 2018; Accepted: 11 November 2018; Published: 16 November 2018

Abstract: Wireless sensor network is an emerging technology, and the collaboration of wireless sensors becomes one of the active research areas for utilizing sensor data. Various sensors collaborate to recognize the changes of a target environment, to identify, if any radical change occurs. For the accuracy improvement, the calibration of sensors has been discussed, and sensor data analytics are becoming popular in research and development. However, they are not satisfactorily efficient for the situations where sensor devices are dynamically moving, abruptly appearing, or disappearing. If the abrupt appearance of sensors is a zero-day attack, and the disappearance of sensors is an ill-functioning comrade, then sensor data analytics of untrusted sensors will result in an indecisive artifact. The predefined sensor requirements or meta-data-based sensor verification is not adaptive to identify dynamically moving sensors. This paper describes a deep-learning approach to verify the trustworthiness of sensors by considering the sensor data only. The proposed verification on sensors can be done without having to use meta-data about sensors or to request consultation from a cloud server. The contribution of this paper includes (1) quality preservation of sensor data for mining analytics. The sensor data are trained to identify their characteristics of outliers: whether they are attack outliers, or outlier-like abrupt changes in environments; and (2) authenticity verification of dynamically moving sensors, which was possible. Previous unknown sensors are also identified by deep-learning approach.

Keywords: sensor collaborations; sensor trustworthiness; dynamic moving-sensor collaboration; sensor calibration

1. Introduction

Wireless sensor network (WSN) is an emerging technology, in part to monitor and detect sensible states of a target environment and, further, to actuate in response to the state change of sensor data. In order to improve the quality of monitoring or detecting the state change of sensor data, sensors are collaborated. The collaboration of sensors becomes one of the active research areas. Various sensors collaborate to monitor the changes of a target environment from different angles, and identify any changes, if they occur. Collaboration can be formed among homogeneous sensors or heterogeneous sensors. For example, in patient rooms, as a patient's breath noise goes up, a camera turns on, and photo images begin to be sent out. This is an example of *heterogeneous sensor collaboration*. Another example is in vehicles, where a tire pressure management system (TPMS) collects air pressure sensor data and turns on an inflated tire sign [1]. This is also an example of *homogeneous sensor collaboration*. Figure 1 illustrates the collaboration of homogeneous sensors in TPMS. As shown in the figure, a four-wheel car has four sensors (APS11–APS14), while an eighteen-wheel truck has 18 sensors (from APS201–APS218). Each sensor sends air pressure data in the microwave frequency bandwidth to the TPMS center server in a car/truck. Note that, according to the data that is to be sent, an appropriate actuation may be taken. This paper does not consider the sensor actuation, but focuses on sensor data verification and intelligent calibration for sensor collaboration.

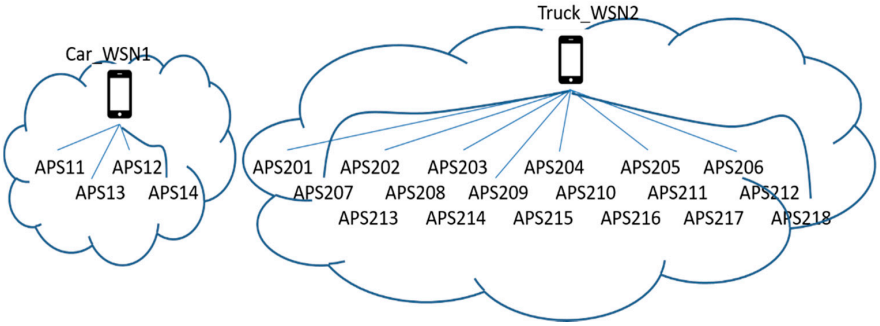


Figure 1. Sensors are deployed to a wireless sensor network (WSN), where APS denotes an air pressure sensor. APSs send sensor data to mobile edge computing.

The sensor collaborations may be able to reach a better decision. Sensor collaboration is valuable for increasing the accuracy of sensor data. For example, in TPMS systems, the decision of tire inflation can be made by checking the PSI (pounds per square inch) of each individual tire, or by verifying the balance amongst the PSI sensors of all tires. If any one of the PSI sensor data is missing, the decision is invalid, unless an error is generated. As such, collaborative analytics of sensor data should verify the authenticity of each and every sensor, so that the outcome of the analytics and corresponding sensor actuations will become valid. Several studies have been published [2–4], but they are not satisfactorily efficient for the situations where sensor devices appear, move, or disappear and, therefore, the location and the identification may not be enlisted. Moreover, some of those moving sensors may be adversarial.

As another example in Figure 2, there are three WSNs deployed. Assume that two of them, WSN2 and AllianceWSN1, are alliance, and AdversaryWSN is adversary. Another assumption is that all the sensors appearing or disappearing are in the same communication protocols, and attackers are, of course, pre-authenticated for data transmission. Sensors are labeled as *S*x, where x denotes a unique number. There may be some abrupt sensor changes: some sensors are not persistently active, and some sensors' IDs keep changing. Sensors are disappearing from one WSN, and entering into another. Based on this figure, consider the following motivating examples.

MOTIVATING EXAMPLE 1 (Sensors Disappearing): Suppose that a noise detector and a camera are deployed in a hospital patient room. If a patient's breath sound becomes heavy and rough, a noise detection sensor can detect it. At some point in time, the sensor turns on the camera to send photos to nurses. However, if the noise detection sensor, e.g., S1 in Figure 2, is dying out, even if patient's condition becomes an emergency, the camera may not be turned on.

Another case of sensors is illustrated in S2. A moving sensor S2 leaves WSN2 for some reason (maybe due to malfunctioning or for transferring purpose). The question raised, in this case, includes how to know whether a sensor disappears and when. As such, how can sensor collaboration be processed without it? What if a cloud server is unavailable to identify the disappearance of sensors? Is there any technique for the remaining sensors to collaborate with no existence of missing sensors or without referring to a cloud server?

MOTIVATING EXAMPLE 2 (Sensors Appearing): Recall again the TPMS system. Suppose that on a highway, cars and trucks are running side by side and, therefore, the sensor data signal of a car's PSI can enter very easily in to a truck's TPMS nearby. This means, as illustrated in Figure 2, sensors are appearing. Sensor S3 appears as it reactivates (maybe since the electric power is resupplied as labeled ③). Sensor S4 is transferred from AllianceWSN1 as labeled ④. Sensors S8 and S9 are entering into WSN2 as labeled ⑤ and ⑥. S8 is transferred clearly from AdversaryWSN3, while S9 enters into WSN2 after drifted from AdversaryWSN3 for a while. The problem raised in this case includes: If there is an adversary sensor attack, how do we know whether a sensor data is untrusted? Can we do it without having to request a consultation from a cloud server?

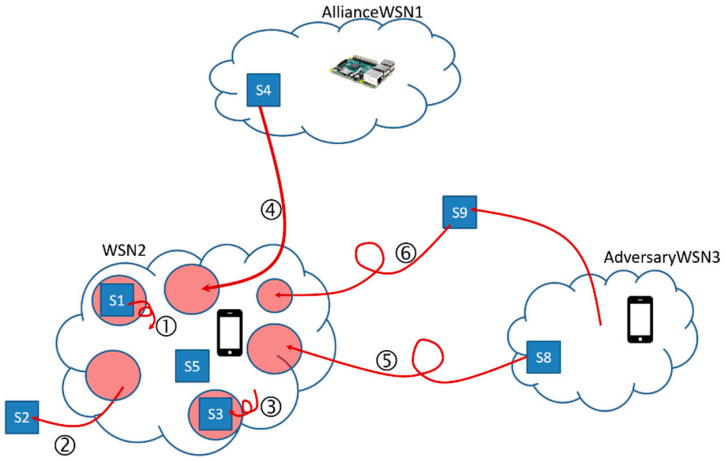


Figure 2. Wireless sensor networks (WSNs) are deployed. Assume two of them, WSN2 and AllianceWSN1, are alliance, while AdversaryWSN3 is adversary. Each sensor, labeled as Sx, is deployed in a specific WSN or in transition.

1.1. Problem Statement

As illustrated in the MOTIVATING EXAMPLEs, sensor devices may participate or disappear unexpectedly in or out of a Fog and Mobile Edge Computing (FMEC) computing environment [1]. Since all participating sensor devices are autonomous, some of them may be attacked, and the attack may be unknown. The issue, here, is how a FMEC computing server knows the trustworthiness of new incoming devices, and how to continually assure the trustworthiness of existing devices.

Another issue may occur on sensor data itself. When sensor data are acquired, or while the data is transmitted to a FMEC computing device, its sensor data may be spoofed, or it may be modified from malfunctioning network channels. How does a FMEC computing server assure the integrity of sensor data?

1.2. Our Approach

To resolve the issues described above, this paper proposes a sensor data-driven sensor device trustworthiness management. When a sensor abruptly appears, or disappears, its trustworthiness can be verified by learning from neighbors. An artificial neural network (aNN) is employed to self-calibration of wireless moving sensors without using any cloud services. With FMEC computing, incoming sensor data is approximated to identify outliers. A sensor device that collects or transmits such outlier data will then be confirmed for its validity. The validity confirmation will be made by a peer group of participating sensor devices.

1.3. Contribution

This paper describes a genetic algorithm to verify the trustworthiness of sensors by considering the sensor data only, but not necessarily using meta-data about sensors or not by consultation from a centralized cloud system. The contribution of this paper includes (1) quality preservation of sensor data for mining analytics and (2) authenticity verification of moving sensors with no external consultation.

1.4. Organization

The remainder of this paper is organized as follows. Section 2 describes preliminaries and related works. Trustworthiness and reputation-based security in sensor nodes is reviewed. Previous works on intelligent sensor data are investigated. The major computing power of devices used for FMEC is also discussed. Section 3 introduces the model of sensors and its representation in JavaScript Object Notation (JSON) [5], which is a de facto standard of data placeholder for wireless transmission. Two classes of JSON are defined: data JSON (or dJSON in short) and reputation assessment JSON

(or rJSON in short). Based on a class, the object of each sensor device is constructed. Section 4 describes an intelligent way of using dJSON to transmit to a FMEC computing device where sensor security is verified, and an rJSON is requested if any sensor device trustworthiness is questioning. The verification of sensor trustworthiness is extended in Section 5. An artificial neural network technique is employed to train sensor appearance and disappearance. To avoid a zero-day attack, the trained patterns will be practiced. Section 6 shows the proposed concept evaluation. For about tens of sensor devices that transmit sensor data, two different types of computing device are evaluated. Section 7 describes the conclusion.

2. Background and Related Work

Major components of FMEC computing include reliable network devices and protocols, computing power, and intelligent software execution to produce services [6]. These FMEC major components are layered in Figure 3, and the top layer of those will be a cloud server.

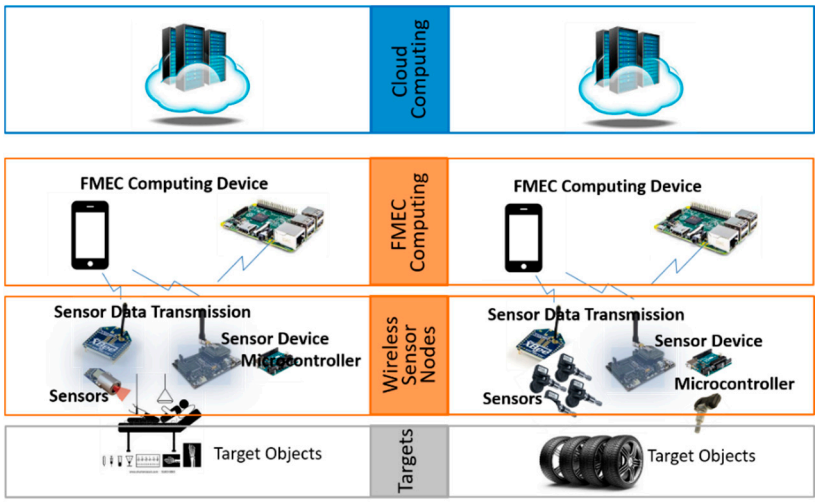


Figure 3. A general architecture of fog and mobile edge computing (FMEC). The FMEC has layers of wireless sensor nodes and computing power, which is placed in between a cloud server and target environments.

It is well known that sensors as a system on chip (SOC) are available on smartphones. Recent smartphone models have a few sophisticated sensors installed and running.

Sensors, if not a system on chip (SOC), are controlled by a microcontroller unit (MCU). If an MCU which holds a sensor moves locations, the sensor is called a mobile sensor. If an MCU, which holds a sensor, sends sensor data wirelessly, the sensor is called a wireless sensor. As sensors are affordable for massive deployment and powerful for monitoring environments, the data acquired by sensors become big and streaming data to analyze. Since the size of sensor data multiplies rapidly and sensor data is streaming at real-time, typical data mining algorithms are not satisfactorily employed for sensor data analytics.

Portable computing powers are available in various devices or microcontroller units (MCUs): BeagleBone [7], Arduino [8], and Raspberry Pi [9], for example. A microcontroller is a single integrated circuit, which consists of a CPU, memory, and programmable IO peripherals.

This section describes each such component with related works.

2.1. Sensor Calibrations and Trustworthiness

Many of the sensors used in healthcare and medical services are traditional medical sensors [2,3,10]: EEG, EMG, ECG, respiration monitor, heart beat counter, etc. The collaboration of these sensor data is very important for health service improvement. One of the critical and active research

areas is environments and context-aware resource allocation in IoT [11]. While the collaboration improves sensor data services, the vulnerability of sensor data will increase.

To improve the functionalities and reliabilities of WSNs, sensors are collaborated. Sensors are collaborated if they are authenticated or trusted. Trustworthiness of collaborative sensors can be verified in several approaches: (1) collaboration by consultation, (2) collaboration by learning from neighbors, and (3) collaboration by artificial neural network (aNN).

Collaboration by consultation. Collaboration sensor authentication can be consulted from many sources, such as cloud moderation, cryptography, or predefined description [12,13]. As illustrated in Figure 2 and motivating examples, both the quality of sensor devices and sensor data should be properly maintained. As such, authentication of sensor devices can be made using the sensor's ID [2,3]. Using sensor IDs is not a good solution for moving sensor devices, due to the list of sensor IDs in one WSN that keeps changing, appearing, or disappearing, etc. The certificates of sensor devices can be used for authentication [4]. Utilization of certificates requires a big overhead and depends on a cloud server.

This approach works well as far as cloud servers are available, and sensors are guaranteed to be active. However, it is not adaptive to dynamical environments where sensors are changing, as illustrated in Figure 2.

Collaboration by learning-from neighbors. The trustworthiness of sensors can be verified by learning from neighbors. Neighbor mobile devices can be discovered by learning spatiotemporal properties of neighbors [14]. Machine learning approaches, e.g., Markov chain theory, are applied to collaborations between neighbor sensors [15]. The sensor collaboration proposed in this paper is to verify the trustworthiness of sensors by learning from neighbors.

Collaboration by aNN. In recent years, artificial or recurrent neural network techniques have been applied to WSNs [16,17]. In this paper, we employ artificial neural network (aNN) to quickly accept or reject, when a new moving sensor approaches to join our WSNs, by learning similar acceptance or rejection cases. Similarly, our aNN is able to learn and practice the patterns of natural dying sensors or those that are attacked and infected.

2.2. Wireless Network Security

IEEE 802.15.4 is a standard for both the physical (PHY) layer and media access control (MAC) layer for low-rate wireless personal area networks (PAN). This standard focuses on low-cost, low-speed ubiquitous communication with a transfer rate of 250 kb/s. The frequency bands used for this protocol include 868.0–868.6 MHz, 902–928 MHz, or 2.4–2.4835 GHz. ZigBee or WirelessHART is an example of the standardized and proprietary networks (or mesh) layer protocols [18].

The IEEE has developed the 802.15.7 standard [19] for short-range communication using visible light. This standard specifies three PHY layers, with support data rate varying from 11.67 kb/s to 96 Mb/s (or even to 120 Mb/s). The frequency bands used for this protocol are between 750 THz and 428 THz, which is harmless to human bodies.

Typically, sensor devices can communicate with smartphones by key pairing using a Bluetooth protocol, and communicate with MCUs by sharing radio pipe addresses [20]. These typical approaches do not protect from malicious sensor attacks, or they do not recognize the issues that might occur in the sensor device side [21,22].

2.3. JSON

JavaScript object notation (JSON) is the most popular de facto standard data format for sending data over wireless communications. A formal definition, a so-called JSON schema, has been proposed [23,24]. Query languages and specifications are also discussed based on JSON [24,25]. As sensor data pieces are collaborated, the trustworthiness of sensor collaborations is not just the matter of sensor devices or sensor data but, also, the trusted communications. In order to achieve the trustworthiness of sensor data communications, sensor data encryption over JSON has been

discussed [26]. The format of data transmission in JSON is in dictionary, where pairs of key and values are listed [5].

3. Sensor Model and JSON Representation

Consider a small world where sensors are deployed, as shown in Figure 3. Sensors are deployed for the target objects. For example, on the left of Figure 3, for patients (as a target), an infrared temperature (as a wireless sensor node) is deployed. On the right TPMS example of the figure, an air pressure sensor (at the wireless sensor node layer) is deployed to measure the tire air (at the target layer). At the FMEC computing layer, mobile edge computing, e.g., mobile smart phones or a little powerful microcontroller, exists to collect sensor data and performs sensor data analytics. Of course, this may be on top of a powerful computing facility, e.g., cloud servers.

As such, sensors are represented in a quadruple ‹*S*,*L*,*T*,*D*›, where *S* denotes sensor types, which include IR sensors, acoustic sensors, thermal sensors, electromagnetic sensors, etc.; *L* denotes location data in a tuple of latitude, longitude, and altitude; *T* denotes temporal data such as the timestamp to acquire sensor data and the timestamp to receive; and *D* denotes sensor data, e.g., amplitude values, phase values, vectorized values, polarized values, etc.

For example, an infrared temperature deployed can be represented in a quadruple:

$$< \text{“IR Temperature Sensor”}, (41.0219,\ -73.8733,\ 19.02), \\ (12:34,\ 12:35),\ 72.09 > \tag{1}$$

which means that the data 72.09 is received at 12:35 from the IR sensor, deployed in an upper state of New York, which transmitted at 12:34.

One or more sensors are deployed, as shown in Figures 1 and 2.

3.1. Sensors and Sensor Hierarchies

As illustrated in MOTIVATING EXAMPLE 1, some sensors are dependent on another. For example, a camera is turned on if a sensor activates. Some sensors may be dependent on another and they may also determine another sensor. There may be three approaches of sensor collaborations to improve decision-making. An activation of one sensor, which we call a *dependent sensor*, may be determined by the outcome of another sensor, which we will call a *determinant sensor*: (1) In order to determine the activation of dependent sensors, each every single data point of determinant sensor data is checked; (2) the tolerance and trigger zones of determinant sensor data are learned at the training phase; and depending on the determinant sensor data, the dependent sensor, can be quickly activated; and (3) no determinant sensor data is used to determine the activation of a dependent sensor. This paper proposes the second approach, and shows the comparison of all three approaches in Section 5.

For example, in Figure 4, a vocal noise sensor, which detects a patient' condition, activates an IR temperature sensor, which can then check the temperature of a patient. This IR temperature sensor can then activate a camera. In the specific context, as illustrated in Figure 4, the vocal noise sensor is a determinant sensor, the IR temperature sensor is a hybrid sensor, and the camera is a dependent sensor in this context. Similarly, recall MOTIVATING EXAMPLE 2. Suppose that an infrared camera is deployed in order to avoid abrupt disappearance or appearance of air pressure sensors. When one of my tires shows high air pressure, an IR camera turns on to see the tires, as well as neighbor's tires. In this case, the air pressure sensor is an independent sensor, and the IR camera sensor is a determinant sensor. Of course, this IR camera sensor may be a hybrid sensor if it activates another sensor further.

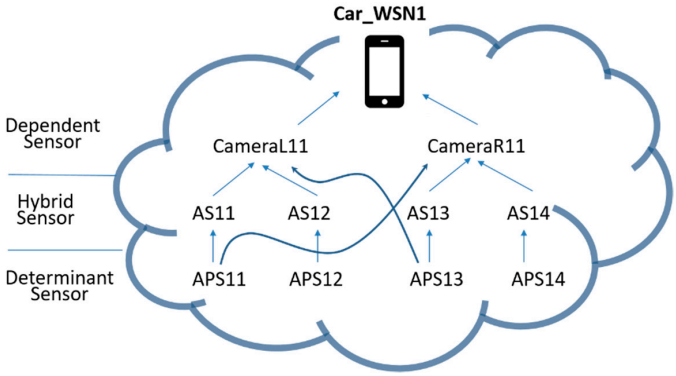


Figure 4. An example of sensor hierarchy and sensor dependence graph.

3.2. Data JSON, dJSON

Sensor data is transmitted in the form of JSON [5], like a dictionary format. Two classes of JSON are defined: data JSON (or dJSON in short) and reputation assessment JSON (or rJSON in short). The dJSON for the above IR temperature sensor data is constructed as follows:

$$\begin{array}{l} \{\ \text{sensorID} : \text{xxyy1}, \\ \text{type} :\ \text{“IR Temperature Sensor”}, \\ \ \text{geocode} :\quad \{\ \text{geoLat} :\ 41.0219, \\ \qquad\qquad\ \text{geoLng} :\ -73.8733, \\ \qquad\qquad\ \ \text{geoAlt} :\ 19.02\ \}, \\ \text{time} :\ \{\ \text{acquire} :\ 12:34, \\ \ \text{arrival} :\ 12:35\ \}, \\ \text{data} :\ [\ 72.09\ ]\ \} \end{array} \tag{2}$$

In the dJSON description above, sensorID identifies a service that is requested by a FMEC computing device, geocode describes the latitude (geoLat), longitude (geoLng), and altitude (geoAlt) of the sensor device, and time describes the time to acquire and the time to be received. Finally, the temperature data acquired is 72.09. For a service, several sensors will be involved to acquire sensor data. For example, a typical patient in a hospital has the fall detection service which collects sensor data from temperature sensors, tilt sensor, liquid monitoring sensors over IV, etc. Note that a geocode in a decimal degree can be converted to hours–minutes–seconds degree, which is very similar to the degree of the time that we consider here. The acquire time, t_{ka}, of sensor k data is likely to be different or very different from the arrival time, t_{kr}, where the arrival time is the time that sensor data arrives at the FMEC computing site.

3.3. Reputation JSON, rJSON

Reputation JSON (in short, rJSON) is JSON data that is described by a sensor device about another sensor device in response to a request from a FMEC computing device. This rJSON contains a key and Boolean value pair: responded?, know_responsible?. Although acquaintance data is considered in this paper, more reputation scoring data can be added depending on the situation and service characteristics. Those two Boolean values can be collected by one of the participating sensor devices, which is designated by the FMEC computing device. The value responded? is determined by the designated sensor device, while know_responsible? is obtained from another peer chosen by the designated sensor.

$$\begin{array}{l} \{\ \text{sensorID} : \text{xxyy2}, \\ \text{Responded?} :\ \text{“NO”}, \\ \text{Know\_responsible?} :\ \text{“YES”}\ \} \end{array} \tag{3}$$

4. Sensor Trustworthiness by Learning from Neighbors

As illustrated in motivating examples, when sensors are abruptly appearing or disappearing in a WSN, verifying the trustworthiness of new entering sensors and the legitimacy of exiting sensors is very important. This section describes how to get a reputation of new entering sensors from neighbors and, therefore, trustworthiness of the new entering sensors.

4.1. Distance Function

Two types of distance measurement are considered in this FMEC computing. Note that, as described above, data JSON *dJSON* carries out geocode, time data, and sensor data collected. For any given two sensors, this paper proposes a way of computing the distances: the location distance, the time distance, and the data distance between two sensors' dJSON. Consider the two sensors, ‹S_1,L_1,T_1,D_1› and ‹S_2,L_2,T_2,D_2›.

$$GeoDist\ (L_i,\ L_j) = \sqrt{(lat_i - lat_j)^2 + (lng_i - lng_j)^2 + (alt_i - alt_j)^2}, \tag{4}$$

$$TimeDist\ (T_i,\ T_j) = \left|(t_{ir} - t_{ia}) - (t_{jr} - t_{ja})\right|, \tag{5}$$

$$DataDist\ (D_i,\ D_j) = \sqrt{(value_i - value_j)^2}, \tag{6}$$

where i and j respectively denotes two sensor devices. *lat*, *lng*, and *alt* are latitude, longitude, and altitude, respectively. t_r and t_a are time to be received at a FMEC computing device and the time to be acquired by a sensor device, respectively. *values* are the sensor data collected respectively by i and j. These datasets are available in dJSON, as shown in (2).

In a small space, e.g., patient ward in a hospital on the left of Figure 3, *Geodist*() will become closer to zero, unless one sensor device is placed on the floor, and the other in the high ceiling. *Geodist*() will be non-negligible if sensors are deployed in a wide area, e.g., in a battle field or in TPMS, as illustrated on the right of Figure 3.

Recall that the temporal data in dJSON, shown in (2), consists of *acquire time*, T_{ka}, at the wireless sensor node site, and *arrival time*, T_{kr}, at FMEC computing site. If *TimeDist* (T_i, T_j) is small, those two sensors when $t_{ir} = t_{jr}$, the two sensors, i and j, are at equi-distance from the FMEC computing site.

The time distance may be used for synchronization of sensors or the tolerance of the time gap between the sensor operations. It means that T_{ia} and T_{ja} can be compared as far as the sensors i and j are synchronized. Sensor data will be compared and analyzed at the FMEC computing site. Since there may be different transmission time delays from different sensors, the computation at the FMEC computing site should be based on acquire time.

4.2. Assessment

In WSNs, data transmission speed (aka, data rate in some context) depends on various factors. For example, the factors include (1) the distance from the location of a wireless sensor node to the location of FMEC computing, and (2) the time elapsed of sensor data transmission. In an ad hoc communication, the data transmission speed varies depending on the carrier signal energy strength and characteristics, such as scattering rate, absorption, attenuation, or interference rate. Due to the media characteristics, absolute speed is indeterministic. This is because in one situation, the same sensor data transmits quickly while, in another situation, it may transmit very slowly. It may be caused by obstacles or air pressure and density from a sensor device to a FMEC computer device. That being said, as far as they are in the similar situation, their speed is acceptable to add them into the alliance sensor set (or trusted sensor set).

In this regard, this paper defines the relative speed by comparing two speeds of different data transmission, each of which is sent out from two different sensor devices. Transmission speed is defined in (7) below. Since there is a time delay from sensors i or j, the data distance, *DataDist*(i, j) takes

the sensor data to be compared not by the arrival time, but by the acquire time. An extended version is *adjDataDist*(*i*, *j*), as shown in (8) below.

$$TransmissionSpeed\ (i,\ j) = \frac{Geodist(L_i - L_j)}{Timedist(T_i - T_j)} \tag{7}$$

$$adjDataDist\ (D_i,\ D_j) = \sqrt{(value_i - value_j)^2}, \text{where } T_{ia} = T_{ja} \tag{8}$$

4.3. Algorithm

The training algorithm shows how WSNs can be trained by knowing the metadata about sensors only. The geolocations and the time elapsed are used to give an efficient assessment with no external consultations.

The learning algorithm is described in Algorithm 1.

Algorithm 1. Learning from Neighbors

This algorithm is to train WSN1 by learning from WSN2. The following parameters are assumed: WSN1 and WSN2 are wireless sensor networks. *i* and *j* are sensors, where *i* is in WSN1 and *j* is in WSN2. TS and US denote alliance and adversary sensor set, respectively. λ, ζ, and δ denotes a threshold for location acceptance, a transmission speed threshold, and a threshold for transmission time delay acceptance, respectively.

Assume |WSN| denotes the number of sensors deployed in WSN.

(1) For each sensor *j* in WSN2
(2) Begin
// Sensor i is the sensor abruptly appearing or disappearing
(3) If (GeoDist(L_i,L_j) < λ) AND (responded? == "YES")
// responded? Available from rJSON
(4) Then If (TransmissionSpeed(*i*,*j*) > ζ) AND (know_responsible? == "NO")
(5) Then If adjDataDist(T_i,T_j) < δ
(6) Then add *j* into TS
(7) Else add *j* into US
(8) Else print(*j*, ": communication error")
(9) Else print(*j*, ": is unable to reach")
(10) End

Note that there is a limited neighbor WSN, where each WSN has limited sensor nodes at a specific point in time. At worst case, this learning algorithm continues until all sensors in all neighbor WSNs are considered. There may be an optimal condition such that this learning phase is satisfactory. However, this paper does not describe that issue.

5. Neural Network Approach to Sensor Trustworthiness

This section describes how sensor data values, geodata, temporal data, etc. of a sensor node, discussed in Section 2, are used in aNN computation. To train WSN1, we consider those from as many sensors as possible from neighbor WSNs. They are formed in an n × m matrix, which can be fed into, together with its weight w_i, which is a random number, and an activation function *a*, e.g., sigmoid or hyperbolic tangent. In Figure 5, the input nodes, s_1, s_2, . . . , s_m are sensor data, while bias threshold B_i and feedback F_i in a certain context are considered optional. The node H_i is computed as $H_i = \sum_{k=0}^{n}(s_i \cdot w_i)$, which will be O_i at the final stage, as shown in the figure.

For simplicity, we normalize the sensor data to be in a minimum bounding rectangle (MBR) for a set of sensor data values that are received for a limited time period. Data inside the MBR will be

1, otherwise 0. Note that this technique can be applied to other data such as geolatitude, longitude, altitude, time to acquire or time to arrive, etc. for aNN computation. The MBR is a fast approximation of incoming sensor data, computed at the FMEC computing layer, which determines whether an incoming sensor data are accepted or not. For example, let s_1 = 95, 98, 95, 100, 98, 99, 95, 96, 99, 97, 100, 98, 102, 98, 95, 96, 99, 97, 98. This sample sensor data s_1 is depicted with five other sensor data in Figure 6. The MBR for $s_1, s_2, \ldots, s_6$, is shown in the figure as well.

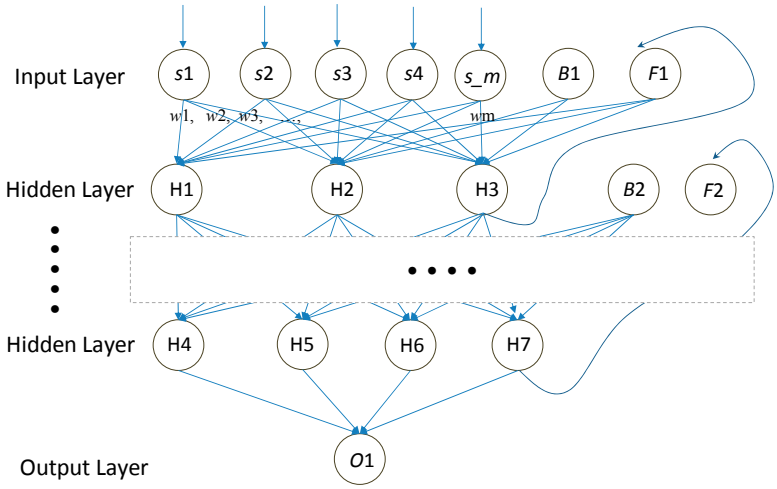


Figure 5. An illustration of an artificial neural network.

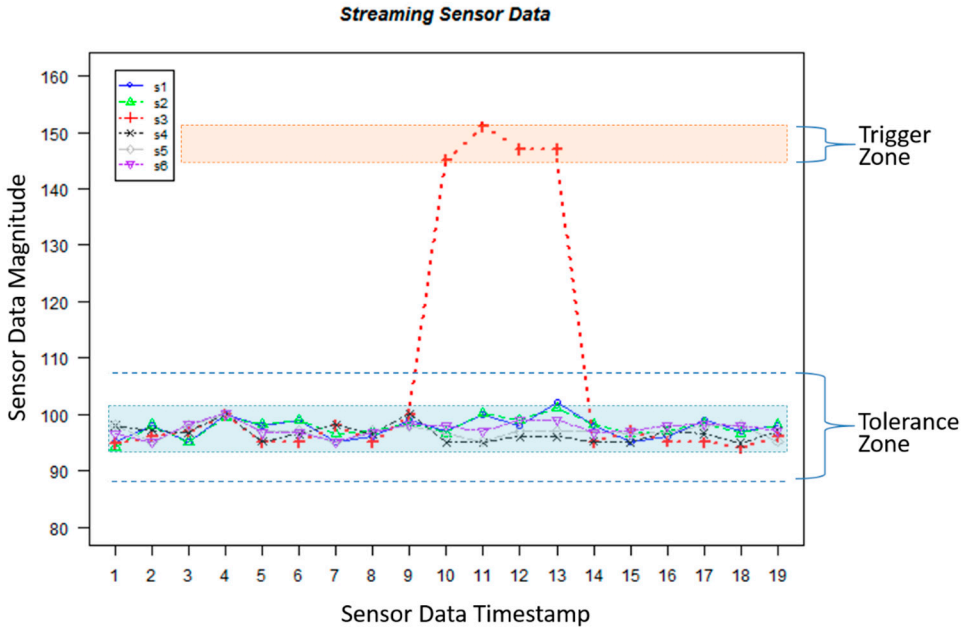


Figure 6. Example of tolerance zone and trigger zone for a sensor dataset.

The aNN trains a WSN by using the given training dataset, which are those in the neighbor WSNs. From the hundreds of thousands of times training from aNN based on the MBR, we propose a tolerance zone, which is defined as the (MBR $\times$ τ), where τ is a parameter $\tau \geq 1$ given based on service and problem domain. If τ = 1, then the MBR of a sample sensor data will be the only data range to be accepted.

In the example in Figure 6, the average is 97.3. If τ is 1.2, which means that 20% more or less the MBR is accepted, the tolerance zone is (78.11, 117.15).

Similarly, we also propose a trigger zone as being yet another MBR of the sensor data that causes an issue of security. As introduced earlier in this paper, the issues of sensor data security include spoofing attack, sensor errors, sensor device errors, etc. An example trigger zone is also shown in Figure 6. There is an issue in sensor data $s3$ in Figure 6, and the trigger zone is (145, 151).

Both the tolerance zone and the trigger zone are obtained from aNN training for a given sample training dataset. There are two phases: training and practicing phases.

Training Phase	**Algorithm 1: Learning from neighbors**
Practicing Phase	For an incoming sensor data, if any one of the following conditions is unsatisfied, trigger an alert. • $GeoDist(si,sj) > \lambda$ for any two sensor datasets, si and sj • $TransmissionSpeed(i,j) < \zeta$ • $adjDataDist(i,j) < \delta$ • Not in Tolerance Zone • But in Trigger Zone

Any data, if out of the tolerant range, will trigger to alert for an additional consideration or action. One of the actions is triggering to collect a reputation flashcard.

6. Experiments

6.1. Implementation Details

There are two major electronic players in FMEC computing: FMEC computing device and sensor device. As discussed, so far, in this paper, there are a few communications and data transactions. This section describes how those electronic players are communicating one another to share which datasets. Consider Figure 7.

Figure 7a illustrates that a few communications are made between a FMEC computing device and a set of sensor devices, as illustrated in TPMS [1] and in Figure 3. Typically, the two steps ① & ② are required to connect between a FMEC device and sensor devices. Once this authentication is made successfully, the device starts to receive ③ sensor data.

In addition to this typical authentication, the FMEC device receives ③ sensor data and data JSON (or dJSON). If the sensor data is in the tolerance zone, the device produces a (whatever it is as defined) service as illustrated in Figure 6. Otherwise, if the sensor data is in the trigger zone, about this suspicious device, the device requests for ④ a reputation JSON (or rJSON) to one of the participating sensor devices. This designated sensor device then takes care of the request to collect rJSON about the suspicious device. Later, the FMEC device ⑤ receives an rJSON, and it decides whether the sensor device is removed from the network or not.

Figure 7b illustrates that a FMEC computing device, particularly a mobile device, performs various missions: training and practicing phases. From the training phase, ❶ tolerance zone and ❷ trigger zone are defined. During practicing, with streaming sensor data, a FMEC device ❸ produces regular services or identifies outliers to trigger an alert. One of the alerts is to request ❹, a reputation request to one of the participating sensor devices. After that, a designated sensor device collects reputations and submit them to the FMEC device.

Figure 7c illustrates that one of the sensor devices, if designated by a FMEC computing device, reports an rJSON. The designated sensor device has two missions: collect a reputation by asking ❷ directly to a suspicious device; and ❸ request an rJSON from another device to collect her own reputation. Collecting all these rJSONs, the designated device ❹ reports to the FMEC computing device. The normal mission of a sensor device is to acquire sensor data and ❶ transmit it to a FMEC computing device.

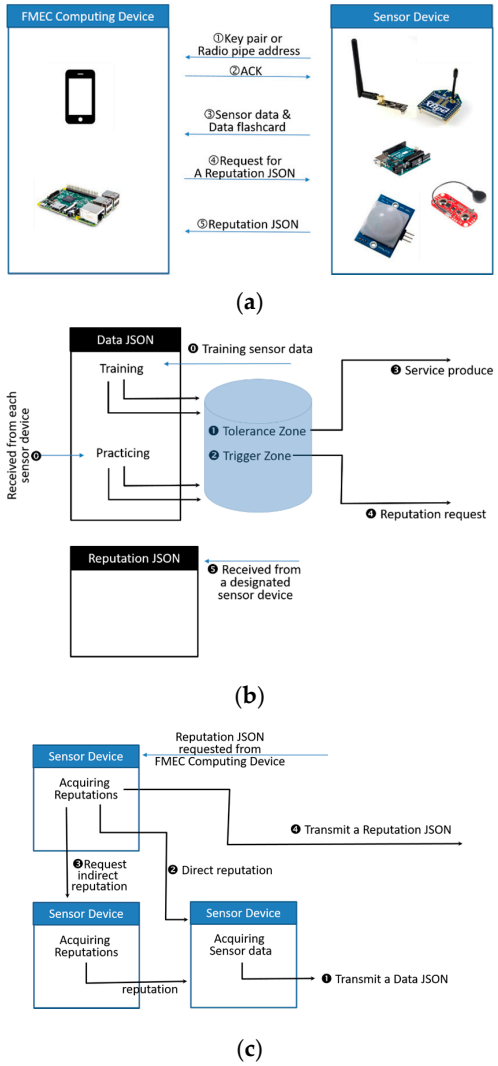


Figure 7. Architecture overview. (**a**) Data and control communication at a global view; (**b**) functions in mobile device; (**c**) functions of sensor devices at sensor node level.

6.2. Triggering from Mobile Devices

In normal cases, such that incoming sensor data falls in the tolerance zone, services are produced at a FMEC computing device, as illustrated in Figure 7b. However, if incoming sensor data have deceits and, so, are not in the trigger zone, then the FMEC device triggers to request for the reputation about the sensor device.

The format of triggering for reputations is as follows:

$$\begin{aligned} &\textbf{WHEN} \ < \text{sensor data } i \text{ in the trigger zone} > \\ &\textbf{IF} \ \text{Geodist}(x,y) \ < \ \lambda \text{ for any sensor data } j \\ &\textbf{AND} \ \text{Timedist}(x,y) \ < \ \omega \\ &\textbf{AND} \ \text{Transmission speed} \ < \ \zeta \\ &\textbf{THEN} \ < \text{request reputation of x to a sensor device j} > \end{aligned} \tag{9}$$

Triggering is activated at practicing time based on the upper/lower-bound obtained at the training time, and its triggering format can be implemented in an iPhone as follows:

Trigger for iPhone (XCode 8 Swift 3 iOS 10)

```
class Practicing {
    var upperbound = Double()
    var lowerbound = Double()
    var sig = [[Double]]()

    init(){}

    init(sigData:[[Double]], upper:Double, lower:Double) {
      self.sig = sigData
      self.upperbound = upper
      self.lowerbound = lower
    }
    func triggerZone() -> [[Int]] {
      var temp = [[Int]]()
      let sigNo = (self.sig).count
      let pointNo = (self.sig[0]).count
      for i in 0..< pointNo {
        for j in 0..< sigNo {
          if self.sig[j][i] > self.upperbound || self.sig[j][i] < self.lowerbound {
            temp.insert([i,j], at:k)
            trigger (position:i, signal:j)
          }}}
      return temp
  }}
```

The above trigger is running as a background daemon process to check the trigger zone for incoming sensor data, and evaluating the conjunction of conditional predicates in (5). If satisfied, the THEN part of (5) will be executed to request the reputation of the suspicious device.

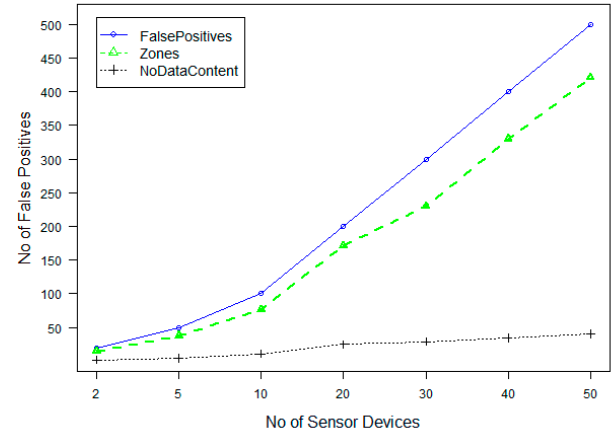
6.3. Results

This section shows the evaluation comparison of the three approaches: (1) checking each and every single data point of sensor data; (2) applying the tolerance and trigger zones learned at the training phase quick to sensor data; and (3) considering no sensor data. The data collected for evaluation is characterized as follows:

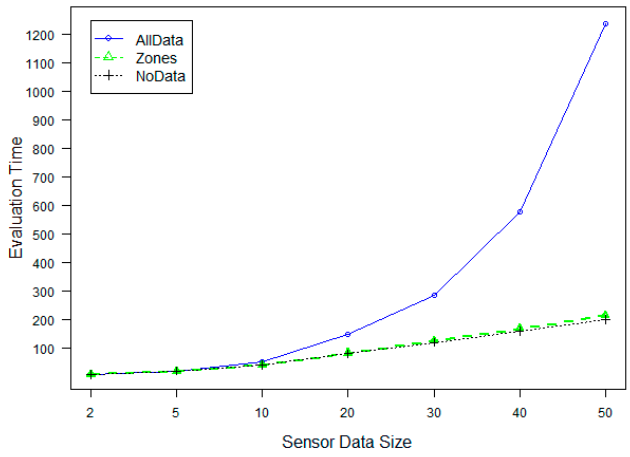
Sensor Devices	No of Data Collections	Duration for Data Collection	Frequency of Sensor Errors	Max. Data
50	30/s	10 min	1/min	1 mill

In this experiment, 50 sensors are deployed, and each such sensor acquires data at every 200 milliseconds. Sensors send their acquired data to the FMEC server for 10 min. The maximum size of the sensor datasets are about 1 million. Assume that a sensor error takes place per minute. Note that sensor errors occur by picking random times for random values beyond the tolerance zone. Not all such random errors are, therefore, in the trigger zone, meaning not all are false positive.

For given sensor data errors or deceits randomly generated, Figure 8a shows that our approach detects about 80% of the errors or deceits. Without our approach, only 1% of the errors may be detected.



(**a**)



(**b**)

Figure 8. Evaluation results. (**a**) Detection of false negatives; (**b**) time for sensor data integrity checking (in milliseconds).

Figure 8b shows the evaluation time elapsed to check the sensor data integrity with respect to sensor data size, i.e., with respect to the number of sensor devices. As shown in Figure 8b, a naïve approach of processing sensor data, in detail, increases rapidly. However, the processing time for the trigger/tolerance zone-based sensor data security assurance remains linear, similar to the case of using no sensor data.

7. Conclusions

This paper described a technique of assuring trustworthiness of sensor devices that may be able to appear or disappear dynamically in WSNs. The technique proposed in this paper is driven by sensor data content, and it can be performed with no external consultation.

Each participating sensor device transmits the sensor data to a FMEC computing device in the JSON format. Neighbor WSNs may be requested to send JSON data about the (acquaintance) reputation on dynamically moving sensors if those sensors have once stayed in the neighboring WSN. By learning from neighbor sensors, a FMEC computing server can check the integrity of sensor data.

An artificial neural network is employed to cope with a zero-day attack. The tolerance and trigger zones are constructed at a training phase and, then, they are used to maintain the trustworthiness of sensors. In this learning method, a sensor dataset can be quick diagnosed to discern the change of

sensor environment from the attack cases of moving sensors. Based on experiments on the TPMS sensor example, the proposed technique enables the positive negatives to be filtered and detected efficiently.

The contribution of the technique proposed in this paper is to verify the integrity of sensor data, which is triggered to assure the trustworthiness of participating sensor devices.

Funding: This research was funded by Mercy College, Faculty Development Grant.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. TPMS. Available online: https://www.waynethomaschevrolet.com/vehicle-warning-lights (accessed on 30 August 2018).
2. Ning, P.; Liu, A.; Du, W. Mitigating DoS attacks against broadcast authentication in wireless sensor networks. *ACM Trans. Sensor Netw.* **2008**, *4*, 1. [CrossRef]
3. Shokri, R.; Poturalski, M.; Ravot, G.; Papadimitratos, P.; Hubaux, J. A practical secure neighbor verification protocol for wireless sensor networks. In Proceedings of the Conference on Wireless Security, Zurich, Switzerland, 16–19 March 2009; pp. 193–200.
4. Dong, Q.; Liu, D.; Ning, P. Providing DoS resistance for signature-based broadcast authentication in sensor networks. *ACM Trans. Embed. Comput. Syst.* **2013**, *12*, 73. [CrossRef]
5. JavaScript Object Notation. Available online: https://www.json.org/ (accessed on 29 August 2018).
6. Iorga, M.; Feldman, L.; Barton, R.; Martin, M.; Goren, N.; Mahmoudi, C. *Fog Computing Conceptual Model, Recommendations of the National Institute of Standards and Technology*; NIST Special Publication 500-325; NIST: Gaithersburg, MD, USA, 2018.
7. BeagleBone Black. Available online: http://beagleboard.org/Products (accessed on 29 August 2018).
8. Arduino. Available online: http://www.arduino.cc/ (accessed on 29 August 2018).
9. Raspberry Pi. Available online: https://www.raspberrypi.org/ (accessed on 29 August 2018).
10. Renuka, N.; Nan, N.; Ismail, W. Embedded RFID tracking system for hospital application using WSN platform. In Proceedings of the IEEE International Conference on RFID-Technologies and Applications, Johor Bahru, Malaysia, 4–5 September 2013; pp. 1–5.
11. Tsiropoulou, E.; Paruchuri, S.; Baras, J. Interest, energy and physical-aware coalition formation and resource allocation in smart IoT applications. In Proceedings of the 51st Annual Conference on Information Sciences and Systems, Baltimore, MD, USA, 22–24 March 2017; pp. 1–6.
12. di Pietro, R.; Soriente, C.; Spognardi, A.; Tsudik, G. Collaborative authentication in unattended WSNs. In Proceedings of the 2nd ACM Conference on Wireless Network Security, Zurich, Switzerland, 16–19 March 2009; pp. 237–244.
13. Cho, M.; Shin, K. ARCMA: Attack-resilient collaborative message authentication in wireless sensor networks. In Proceedings of the 4th Conference on Security and Privacy in Communication Networks, Istanbul, Turkey, 22–25 September 2008.
14. Guo, L.C.S.; Shu, Y.; Zhang, F.; Gu, Y.; Chen, J.; He, T. Selective reference mechanism for neighbor discovery in low-duty wireless sensor networks. In Proceedings of the 9th Conference on Embedded Networked Sensor Systems, Seattle, WA, USA, 1–14 November 2011; pp. 367–368.
15. Ji, X.; Hou, Y.; Hou, C. A distributed incremental learning method for sparse kernel machine over wireless network. In Proceedings of the 2017 Conference on Machine Learning and Soft Computing, Ho Chi Minh City, Vietnam, 13–16 January 2017; pp. 181–186.
16. Zhao, H.; Hua, Q.; Chen, H.; Ye, Y.; Wang, H.; Tan, S.; Tlelo-cuautle, E. Thermal-sensor-based occupancy detection for smart buildings using machine-learning methods. *ACM Trans. Des. Autom. Electron. Syst.* **2018**, *23*, 54. [CrossRef]
17. Wu, P.; Tu, Y.; Yang, Z.; Jatowt, A.; Odagaki, M. Deep modeling of the evolution of user preferences and item attributes in dynamic social networks. In Proceedings of the Conference on WWW, Lyon, France, 23–27 April 2018; pp. 115–116.
18. AutomationNetworkSelection_3rdEd_Chapter3. Available online: https://www.isa.org/pdfs/automation-network-3rd-edition-chapter-3/ (accessed on 29 August 2018).

19. IEEE. *IEEE Standard for Local and Metropolitan Area Networks (Part 15.7) Short-Range Wireless Optical Communication Using Visible Light*; IEEE Std 802.15.7-2011; IEEE: Piscataway, NJ, USA, 2011.
20. Sun, D.; Mu, Y.; Susilo, W. Man-in-the-middle attacks on secure simple pairing in Bluetooth standard v5.0 and its countermeasure. *Pers. Ubiquitous Comput.* **2018**, *22*, 55–67. [CrossRef]
21. Alanezi, K.; Rafiq, R.; Chen, L.; Mishra, S. Leveraging BLE and social trust to enable mobile in situ collaborations. In Proceedings of the ACM Conference on Ubiquitous Information Management and Communication, Beppu, Japan, 5–7 January 2017.
22. Chang, C.; Alexandris, K.; Nikaein, N. MEC architectural implications for LTE/LTE-A networks. In Proceedings of the ACM Workshop on Mobility in the Evolving Internet Architecture, New York, NY, USA, 3–7 October 2016; pp. 13–18.
23. Pezoa, F.; Reutter, J.; Suarez, F. Foundations of JSON schema. In Proceedings of the WWW Conference, Montreal, QC, Canada, 11–15 April 2016; pp. 263–273.
24. Bourhis, P.; Reutter, J.; Suarez, F.; Vrgoc, D. JSON: Data model, query languages and schema specification. In Proceedings of the ACM PODS Conference, Chicago, IL, USA, 14–19 May 2017; pp. 123–135.
25. Isena, P.L.; Troncy, R. Transformaing the JSON output of SPARQL queries for linked data clients. In Proceedings of the WWW Conference, Lyon, France, 23–27 April 2018; pp. 775–779.
26. Detering, D.; Somorovsky, J.; Mainka, C. On the (in-)security of javascript object signing and encryuption. In Proceedings of the 1st Reversing and Offensive-Oriented Trends Symposium, Vienna, Austria, 16–17 November 2017.