A SECURE ARCHITECTURE FOR RESOURCE CONSTRAINED IOT ENVIRONMENT

Omar Farooq[*a], Parminder Singh[b]

[a]*Department of Computer Science and Engineering, Lovely Professional University, Jalandhar, India;*

[b]*Department of Computer Science and Engineering, Lovely Professional University, Jalandhar, India*

majid2432@gmail.com, parminder.16479@lpu.co.in

Abstract: *One of the most exciting emerging concepts nowadays is the Internet of Things. However, digital currency has run into issues with how quickly it has been adopted. The number of IoT devices is increasing exponentially, and presently we have more than 20000 million objects connected to the network. The amount of data and complexity circulating across networks is also growing exponentially. IoT plays a measure role in this growth rate of IoT data traffic, resulting in a significant rise in data traffic reaching the cloud or data centers. The response time of IoT systems is affected by the growth of data traffic as this may not be appropriate for sensitive environments. This paper presents a framework and a machine learning approach for the data management of IoT edge-cloud environments with resource-constrained IoT applications. In this paper, the security aspect has also been discussed for the resource-constrained IoT framework.*

Keywords: **IoT, Resource Management, Machine Learning, Data Classification, Security**

1. INTRODUCTION

The Internet of Things (IoT) is a technology that strongly enters people's reality. All environments are involved, urban, industrial, office, or home. The speed of adoption of the technology has produced a certain disorder and informality in the process. As a consequence, important elements were left aside; one of the most relevant is that of security [1].

In principle, IoT security does not have to be different from security in a typical computer network. In practice, however, there are environmental difficulties that further complicate the security problem. Many IoT devices are computationally limited, preventing the use of several known robust security mechanisms. The large number of devices that can be involved in an IoT network and the exponential increase in the number of interactions exacerbate the problem. The diversity of the equipment used, both in hardware and software, complicates the possibility of generalizing the proposed solutions [2]. There is a wide variety of methods and tools that can be used to undertake the work [3]. In the IoT environment, we have resource constraints in most of the scenarios. Therefore, including the security aspect in the IoT framework has to be in accordance with the processing power, battery life, communication range, and other characteristics of the resource constraints IoT framework.

The proposed Framework distributes the processing load to the last nodes of a digital network (sensors in the case of IoT). The use of computing type poses very attractive advantages for IoT solution providers. For example, they allow to minimize latency and preserve network bandwidth, operate reliably, speeding up decision-making, capture and protect a large number and types of data, and transfer the data to the most appropriate place for processing, with better analysis of local data. Edge computing technologies have been on the rise for several years, but the reach of IoT technology is accelerating its take-off process. As for the factors driving this change, two stand out: Falling prices for peripheral devices with increasing processing power. Centralized infrastructures support the increasing workload. Edge computing technology also arrives at artificial intelligence on devices much more feasible. It allows companies to leverage their data series in real-time rather than working with terabytes of data in central repositories in the world real-time cloud. In the next few years or decades, the technology may evolve to find a balance point between the cloud and more powerful distributed edge devices. Software vendors are developing specific, more robust, and secure infrastructures and security solutions. Providers will begin to incorporate security solutions for peripheral components into their current service offering to prevent data loss, provide network health diagnostics, and protect against threats.

This paper also discusses different security protocols for the resource-constrained IoT framework. The rest of the paper is organized as Literature Review, IoT Ecosystem, Secure Architecture, Machine Learning, Result and Analysis and Conclusion.

2. BACKGROUND

In the ecosystem of IoT devices, these are largely unsafe, as they are small and energy-efficient devices; therefore, they also have limited computational resources. This last condition affects a lot when trying to include complex security schemes [8]. At an industrial level, several enterprises are applying IoT, for example, in intelligent transport [9, 10] and agriculture [11]; however, one of the great current home and office automation goals is connected living. This objective requires important advances in the field of IoT, where it is necessary to provide answers to the problems related to the enormous increase in devices that must interact [12]. A particular case in the IoT is that of smart homes since many times the solutions implemented are ad-hoc by the users themselves, who usually try to reduce costs and efforts as much as possible, which is generally reflected in a minimal and probably non-existent security scheme [13].

The devices involved require interconnection in a many-to-many scheme. It is necessary to implement an identity management system to ensure the exchange of information that scales appropriately. In this sense, [14] proposes a home system that combines EAP, OAuth, and DTLS. Also concerned with identity management and access control, [15] confirm the possibilities of OAuth and make an architectural proposal compatible with services.

Analyze the problem of IoT security [16, 17] in the home and highlight how important it is to prevent sensors from indiscriminately capturing and distributing household data. As an example, they present the case of private conversations, which should not be published. Among the possible approaches, they mention a viable alternative that is oriented to services, to balance between centralization and distribution of control. Furthermore, the trend in software and distributed applications seems to be generally directed towards the use of microservices [18, 19], delving along this line. They highlight the benefits that a microservices architecture, based on TLS / PKI, can have in IoT by lightening development and maintenance tasks, beneficial both for suppliers and distributors and for users while reinforcing interconnection security. Case studies such as that of [20, 21] confirm the possibilities of these techniques in a practical way.

Said work then analyzes the possibilities of using SSH and highlights the advantages provided by the data compression issue included in the said protocol, which is especially advantageous when working over HTTP. Although most of the related work's SSL / TLS is the preferred security and encryption mechanism, what [22] proposes is very interesting when analyzing the complications at a practical level in IoT with TLS. Contribute to the IoT environment with a model-managed approach and propose an OAuth-oriented model with a strong UML inclination [23].

Another interesting architectural and security proposal is presented by [24], which the SSL certification authorities; an approach of local certification authorities would be used, which more frequently, but also with a lighter process, would authenticate the IoT equipment. This proposal, through transformations, could be adapted to a specific architecture, offering the possibility of customizing it to the required environment. It can be considered that a middle point between the two proposals would be that of [25], which uses traditional certificates, but with close authentication, rather than at the node level; they emphasize that this mechanism could be complemented with one of authorization, such as OAuth or similar.

Emphasize OAuth [24], but above all, with the particularity of concentrating its security architecture on the gateway equipment, where the base station or sink node resides, which is in charge of processing heavy of authenticating, authorizing, and establishing the links between clients and resources. This approach is very relevant when we consider how susceptible those edge devices are linked to edge computing [25], through which an entire IoT system can be compromised. One of the high-security points in edge systems in IoT is usually related to MQTT (or similar protocols). Several works, such as that of propose improvements over said protocol.

3. IOT ECOSYSTEM

This work mainly considered that within the IoT ecosystem, it is necessary to segment the location, scope, and access of the equipment involved in two layers: local or edge and centralized. In the local layer, represented schematically in Figure 1, we have those elements that will invariably be installed in the smart home or office, such as sensors, actuators, edge processors such as gateways and brokers, and mobile user devices.

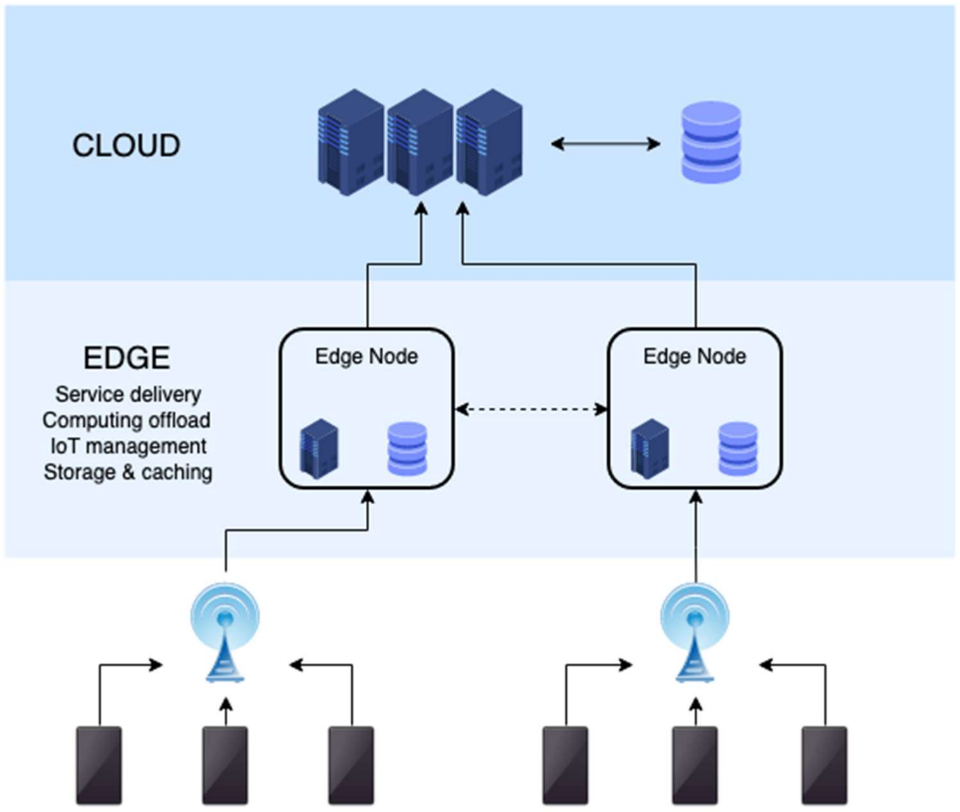

Figure 1: Components at the local or edge layer of the IoT system

Sensors are all equipment capable of capturing physical phenomena, virtual events, or periodic signals. Those sensors with the necessary capacity can communicate directly with the central broker; otherwise, they will interact with equipment in the edge processing subsystem. The sensors will be static when they emit a constant signal that would generally be used by mobile equipment moving in the environment as Bluetooth beacons for positioning. The dynamic sensors will capture measurements of the environment, which will vary depending on the environmental conditions, such as luminescence, temperature, humidity, among others.

The actuators will allow interaction with hardware or software generating events or actions. They will receive instructions either directly from the broker or a preprocessor. They are divided into premises located in the intelligent environment, such as light or temperature controllers. They can also be remote, such as those capable of sending instructions, probably over the network, to a distant computer, but controlled from the home or smart offices, such as when it is required to send an SMS, email, or tweet.

Finally, this layer contemplates the preprocessor equipment, which can also be called edge processors or brokers. These computers can be Raspberri Pi or small computers such as tablets. These capture raw data from the sensors to forward it to the centralized broker or actuator when the sensor is incapable. The information can be sent as received from the sensor, or it can be pre-processed, and this result sent. In the centralized layer, presented in Figure 2, the teams in charge of the general coordination of all the components are necessary, i.e., administration, processing, and persistence. These subsystems are linked to each other and also to the edge layer through a centralized broker.

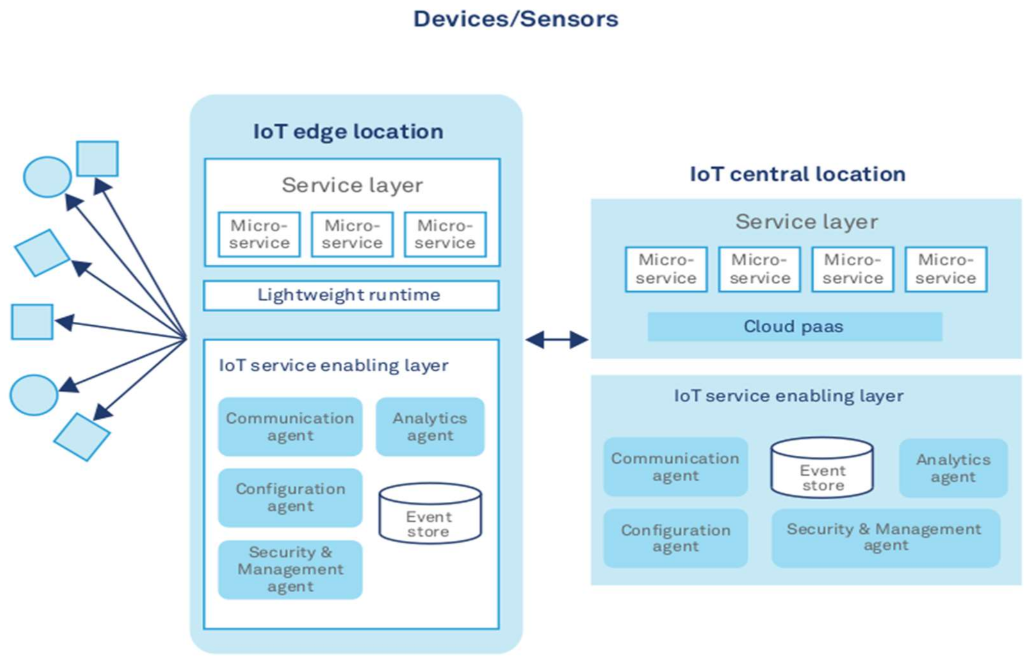

Figure 2: Components in the centralized layer of the IoT system

The administration subsystem defines the parameters and configuration of the system presents the web interfaces for the administrator users to interact with the entire system. In the case where central processing is done with multiple machines, it also manages the resulting cluster. The main part of this subsystem is then monitoring, which will allow all types of users to review the relevant information, preferably through dashboards and statistical tables. For all heavy information processing, the corresponding subsystem takes the data collected from the broker and processes it as defined by the specific applications or needs of the IoT system.

In general, the processing will be divided depending, above all, on the urgency of the processing, in real-time, which processes the data in a continuous flow, as they arrive from the sensors; in memory, which collects the information in the cluster's memory, depending on the needs, and processes it in small batches; and batch, which interacts in general with the storage system, for processes where the amount of data is greater than what fits in the system's live memory. The information generated by the IoT system is directed to the persistence module, which safeguards the data for later use either in the development of models or in the generation of reports. Several alternatives must be considered, depending on the size of the information and the way it will be accessed.

Finally, the entire system, and more specifically the border and centralized layers, must connect and exchange information, which is achieved through a central broker, in charge of managing all the message queues, thus reducing the complexity of the interactions.

4. SECURE ARCHITECTURE

The resulting architectural design took into consideration, above all, the need to ensure the exchange of information of all the components of the system, taking care of the speed of calculation at all times. These elements require reconciling characteristics that are often incompatible. For example, more robust cryptography systems may require more computing power than many lightweight devices, such as sensors, provide. The final architecture designed, implemented and tested is the one outlined in Figure 3, which will be described in detail below. In the first place, the components involved will be specified, and then the security functionality in general will be presented.

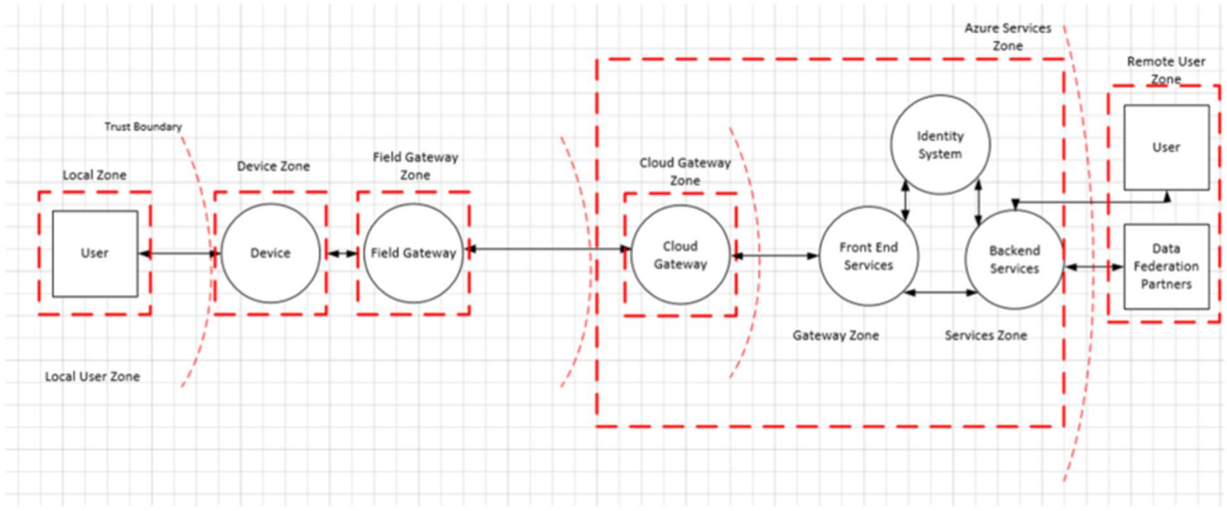

Figure 3: Secure Architecture

4.1. Components

In this section we will try to define in a general way the types of components or equipment involved in the architecture proposed in Figure 3, in which there are basically those in charge of the registry service (Registry), the equipment providing authentication services for clients and users (UAA) and then, in a broad way, all the teams providing general services, as well as all the client teams (Services and Clients). Although for simplicity the components will be referred to in the singular, the architecture considers that, for each category or type of component, especially services, they can work in clusters.

4.1.1. Registry Services (REG)

The main task of the REG is to allow services to register through IP and aliases (service name) and thus make them available to customers, who will connect to the REG to request the information with which they will finally connect to the services of interest. The REG also provides a load balancing service, by detecting that a service is registered in a cluster (multiple computers with the same service). The first point of contact for all other components of the system, whether these services or clients, is the REG, which requires a static IP; all the other components of the system, however, can work with dynamic IPs, via DNS.

4.1.2. Authentication services (UAA)

The UAA takes its acronym from the English User Authentication and Authorization. Within our architecture it basically provides us with the authentication service, which works under OAuth2. The UAA stores the data of all the clients in the system, including their roles; With this information, the UAA user services may or may not authorize the use of certain elements. Any component can connect with the UAA to, through its client credentials (user and password), request an access token. In the same way, any component of the system can request the UAA to validate a token received from a third party.

4.1.3. Generic Services (SRV) and Client Computers (CLI)

The last category of components accommodates all other services and all clients. In general, these components will interact with each other after having registered / authenticated in the system with the help of the REG and UAA. The services can be very diverse and it is up to the system administrator to decide which ones to require. However, in our architecture for IoT, there are some that are fundamental, for which they have been implemented in the test system, and they will be mentioned below. To allow interconnectivity and, at the same time, reduce its complexity, the messaging service was implemented, which in the methodology is represented by the central broker.

The broker is able to receive and distribute all the messages circulating in the system and basically allows all services and clients to establish a single connection with the broker in general, to deposit messages and retrieve them from one or more queues. This broker can work with any communication protocol, or a combination of them. However, since in the world of IoT, at least today, the most widespread protocol is the Message Queuing Telemetry Transport (MQTT), it is the one used in the implementation presented here. Another service implemented for the proof of concept of the architecture is the one related to persistence, as a necessary support to subsequently implement batch processing.

For this, a transit service was implemented that takes the information from the broker and transfers it to a Hadoop cluster, where different types of tools of said ecosystem can be used to process the information. One of the cases that was worked on, given the nature of the IoT information, especially that from the sensors, was that of time series. For this, two data series services were built, thus providing graphing and trend analysis, among others. Regarding customers, all the sensors are considered here, which provide information to the system, the actuators, which react with the environment thanks to the information from the system, and all those devices, mobile or desktop, that allow the user to enter to configure the system, collect processed information, or even act also as sensors and actuators.

4.2. Security Schemes

The system's security architecture comprises three fundamental scenarios: the basic one, to which all elements must adhere to in their transactions, unless otherwise specified; the lightweight scheme, generally used only when starting a worker process on the system; and the strengthened one, for relationships of trust between services.

4.2.1. Basic scheme

This is the default scheme that the system components will use in their transactions. This scheme is represented in Figure 3 by the dotted line that encompasses the system, and uses a combination of one-way TLS plus OAuth2. Every service must provide its public security certificate (PKI) to clients, who can then validate it with the certificate authority (CA). Also, every client must provide the services with an OAuth access token so that they can validate it with the authentication service. The use of TLS, in our scheme, is especially necessary to be able to encrypt the content of the information that is transmitted. It is used only on the services side to limit as much as possible the overhead that would imply, above all, at the administration level (but also of resources and processing), use it in all the components. The security breach that appears is compensated with the use of OAuth2, through which the clients are in turn validated by the services.

4.2.2. Light outline

This is a scheme that could be considered insecure, which is why it is provided only for those cases where an access token cannot yet be obtained, or when it is considered redundant to request it. Two cases exist at the moment in the work environment, which implement this scheme. When components enter the system in order to initiate their transactions, it is generally necessary to have the OAuth token, but since the UAA server IP may have changed, the first step is to contact REG to request the updated IP. For this connection, the client does not yet have the token, which is why it is not possible to work with the basic scheme. The second implementation of this scheme was applied to avoid unnecessary redundant connections and occurs when the service receives the token and must validate it with the UAA.

It is for this type of case that the lightweight scheme comes into play. The service provides its PKI with which the communication is encrypted, but the client is not obliged to validate it (although it is recommended that they do so), the service provides an "insecure" access point for the client, which does not require the token. This is the scheme provided by REG exclusively to be able to deliver the UAA data.

4.2.3. Strengthened Schema

Similar to the problem worked in the light scheme, sometimes two services require interconnection, but at least one of them (who acts as a client) is unable to obtain its access token. When dealing with services, it is not convenient to open an insecure channel as is done in the lightweight scheme. In order to maintain the security standard, then, it was decided to implement a two-way TLS scheme, which is possible, without incurring greater overhead, since, being services, they already have their PKI anyway. Additionally, in general, the services will be executed in equipment with greater processing capacity. This implementation also requires a dedicated channel to be able to execute this type of validation and the example is given by the communication between the REG and the UAA. The UAA is the one that provides the access tokens and therefore should validate itself which would generate a security hole. The REG, then, opens a dedicated channel so that a UAA service can register in this way at all times. By connecting the UAA with the REG, they exchange their respective PKIs, mutually validating each other over TLS, without reducing the system's security standard.

4.3. Functionality

Returning to Figure 3, the dotted line represents the scope of the basic security scheme, which encompasses the entire system. Internally, the numbers 1, 2 and 3 can be seen, circled, which indicate the recommended starting order to guarantee the fluidity of the service. In practice, at least the services have in their base library the functionality to retry the connection when this starting order is not respected. However, this can be subject to unnecessary delay.

In the first place, the registration server, REG, is started, which will provide a central access point for the acquisition of contact information for the other services: every service will register in the REG its respective IP and its alias (service name) and every client will search here, by aliases, for the IP of the service required to be able to connect with it. REG offers three access points, each of which must handle a different security mode: the first, lightweight, allows any client to obtain the UAA's IP without any additional security; the second mode, strengthened, allows the connection of the UAA using two-way TLS; the last, basic one, which requires OAuth2, allows clients to request information from services, and from services to record their contact information.

Second, an Authentication Server (UAA) is started, which will provide OAuth2 credentials to clients. The enhanced security mechanism, with two-way TLS validation, is used between the UAA and the REG. The UAA connects with the REG, as well as any other service, to give it its IP and aliases and thus be available to the entire system. Once these two services, REG and UAA, are online, all the rest of the components, services and clients can start their work.

Finally, then, as point 3, any other component, be it this service or client, will proceed as follows: first, using the lightweight security scheme, they will connect with the REG to request the IP of the UAA; They establish the connection with the UAA and request the access token, using their client credentials. With the access token in hand, the basic security scheme can already be used and, in the case of services, they will be registered with the REG, delivering IP and aliases, to wait for client requests, or act as a client from another service, as needed. In the case of a client, the next step is to use a service, where the basic security scheme will be applied; it connects to the REG and by means of an alias it requests the IP of the service of interest, and then connects with said service.

5. MACHINE LEARNING

To perform the object detection, we will use the K-NN algorithm, the purpose of the K-NN algorithm is to use a database in which the data points are separated into several classes to predict the classification of a new sample point. Therefore, K-NN could and probably should be one of the first choices for a classification study when there is little or no prior knowledge about the distribution data. The K-NN algorithm is based on the similarity of characteristics: the way in which the out-of-sample characteristics resemble our training set determines how we classify a given data point; we will present the operation of the algorithm:

Steps:

The test sample (green circle) must be classified in the first class of blue squares or in the second class of red triangles. If k = 3 (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle.

If k = 5 (dotted line circle) it is assigned to the first class (3 squares versus 2 triangles inside the outer circle) figure 4.

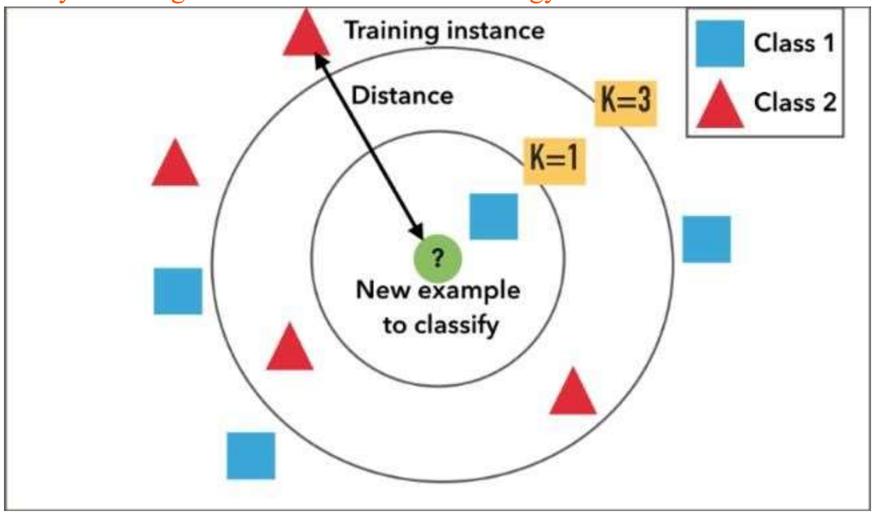

Figure 4: Operation of the Algorithm

K-NN can be used for classification: the result is a class membership (predicts a class, a discrete value). An object is classified by the majority vote of its neighbors, assigning the object to the most common class among its k closest neighbors. It can also be used for regression: the output is the value for the object (predicts continuous values). This value is the average (or median) of the values of its k closest neighbors.

Some other features of K-NN:

- K-NN stores the entire training data set that it uses as a proxy.
- K-NN does not learn any models.
- K-NN makes just-in-time predictions by calculating the similarity between an input sample and each training instance.

5.1. Deployment and Testing

The implementation decision was mainly that each of the components can be executed on a variety of computers and with the least interdependence, for which we proceeded to work on a microservices architecture that allows their deployment either as independent processes, or within a containerization structure, such as Docker. For desktop services and clients Java was used with Spring Boot in general. For the central messaging service, it was decided to work using the MQTT protocol and for the development of the broker the Moquette library [22] was taken as a base, to which it was modified to add support for OAuth2 mainly. An HDFS cluster was used as a basis for persistence services [23] time series services were built on the local network, which, according to the current interests of the paper, were the most suitable for processing the data coming from the sensors.

5.2. Algorithm for Proposed Work:

Let (X_i, C_i) where i = 1, 2……., n be data points. X_i denotes feature values & C_i denotes labels for X_i for each i.
Assuming the number of classes as 'c'
$C_i \in \{1, 2, 3, \ldots\ldots, c\}$ for all values of i

Let x be a point for which label is not known, and we would like to find the label class using k-nearest neighbour algorithms.

1. Calculate "d (x, x_i)" i =1, 2, ….., **n**; where **d** denotes the Euclidean distance between the points.
2. Arrange the calculated **n** Euclidean distances in non-decreasing order.
3. Let **k** be a +ve integer, take the first **k** distances from this sorted list.
4. Find those **k**-points corresponding to these **k**-distances.
5. Let $\mathbf{k}_i$ denotes the number of points belonging to the ith class among **k** points i.e., $k \geq 0$
6. If $k_i > k_j \ \forall \ i \neq j$ then put x in class i.

5.3. Setup:

It interacted with OpenTSDB and Prometheus As for customers, it was decided to build generic libraries for different types of systems, which facilitate the process of developing specific applications. A Java client library was developed, one for Android mobiles, one for Arduino MKR and another for ESP32, the latter three based on Eclipse Paho. The Arduino libraries were intended exclusively for use in sensor and actuator controllers. System tests were conducted in a controlled office environment. The main equipment was an RPi 3B + that served as a Wi-Fi Gateway, providing DNS and NTP services, among others. MKR 1010 and ESP32 controllers were used simultaneously, which permanently received information from temperature, humidity and noise sensors, such as the DHT11 and the KY038. The RPi3B + also hosted the REG, UAA and messaging broker MQTT services. Persistence services over HDFS as well as TSDB were installed on Linux on an i5-4210U with 8 GB of RAM. In this last equipment, generic services were also installed for sending and receiving messages, with which the fluidity of the interaction was verified.

As a mobile client, a N9005 was used, which had two functions: a dummy sensor, sending a large number of random numbers to the system, and a light actuator, warning each time that the measurements of the real sensors exceeded certain levels parameterized in the app. In Figure 11 is presented one of the setup of the testing system.

In the experimentation related to the security tests, it was decided to assume that a possible attacker was already connected to the local network and that he was, potentially, capable of making any request and capturing all the traffic. In the first test, Zap was used to perform both active and passive scanning, and it was verified that security was maintained (encrypted traffic), except in the case (documented in the architecture) of the lightweight scheme.

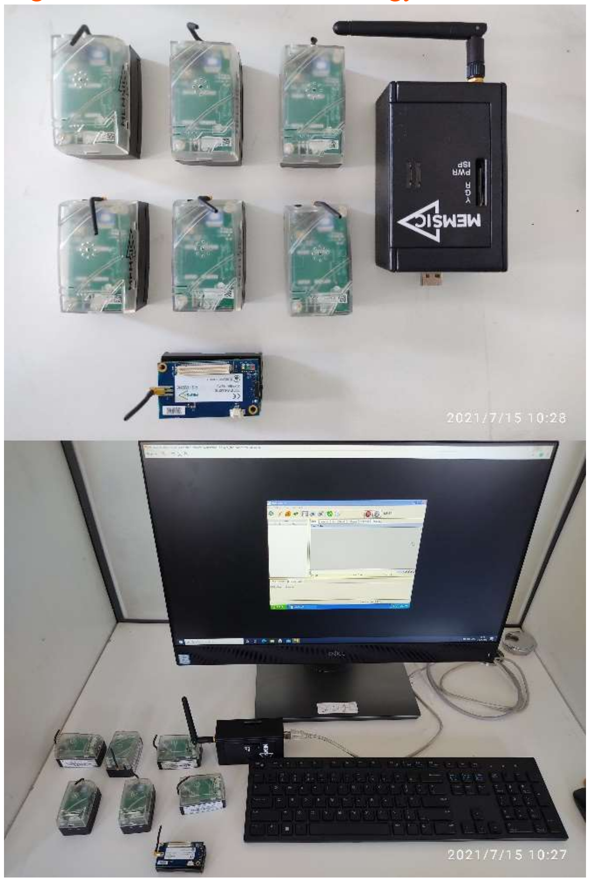

Figure 5: Setup

In Figure 5 from left to right, a cluster of 3 Raspberry Pi is observed running all the services on Docker; an ESP8266 monitoring noise level with a KY038; a MKR1010 monitoring room temperature with a DHT11; an N9005 injecting random messages, and a desktop monitoring all services.

6. RESULTS AND ANALYSIS

The process of including complex security schemes in lightweight IoT devices is not trivial as stated by Khan [14] and in this work this statement is agreed. Two notable cases were that the handling of TLS with auto-generated certificates for MKR 1010 required regenerating all the firmware to include the CA, and that it could not be carried out on ESP8266 controllers due to the unavailability, in practice, of open-source libraries. This is sufficiently complete to guarantee the expected level of security. The approach adopted generally takes up the warnings made by Lin and Bergmann tries to provide a sufficiently complete and simple system so that with minimal technical support it could be implemented at home, and then managed directly by the user.

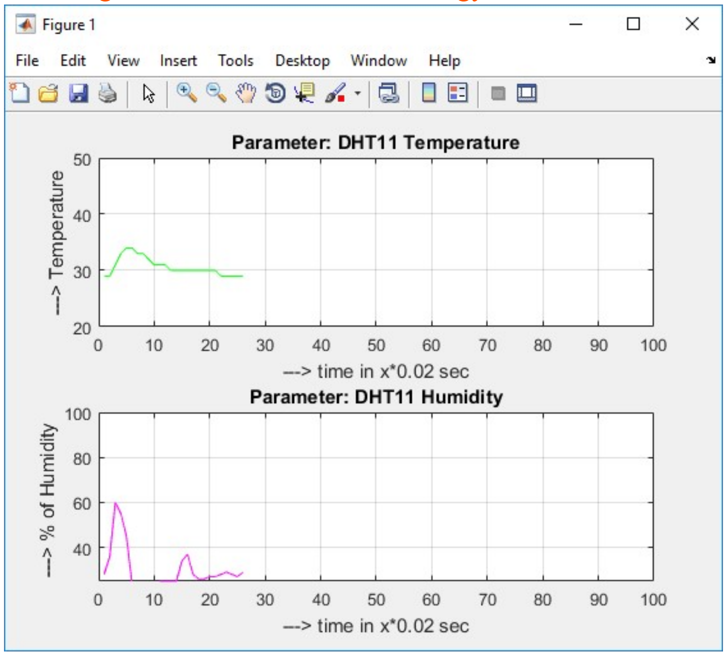

Figure 7(a): Measurement of Effect of change in Temp/ Humidity in DTH11 Sensor

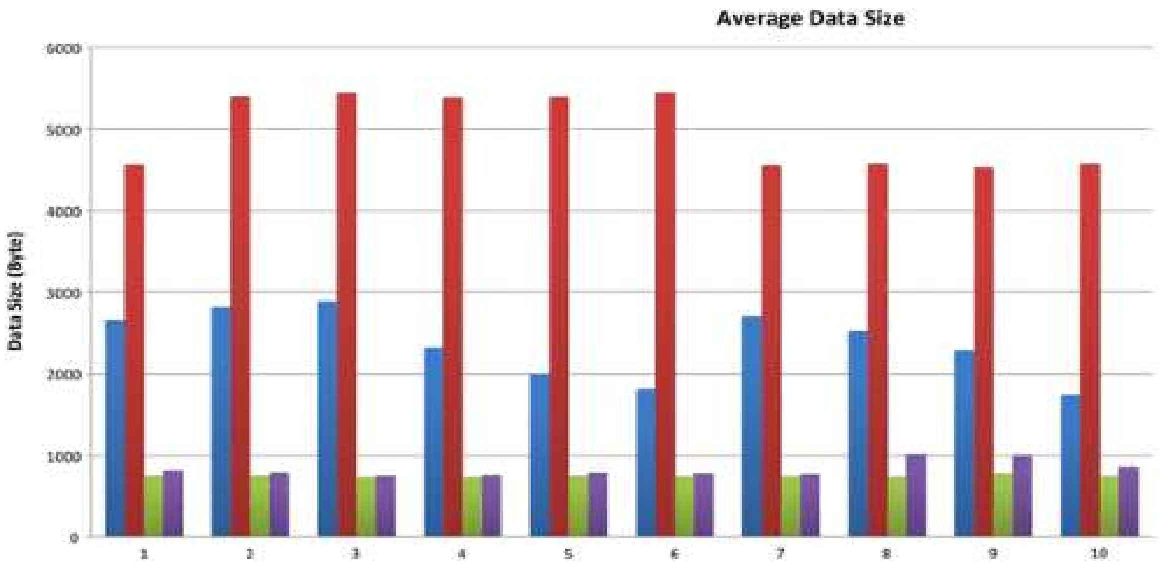

Figure 7(b): Measurement of Effect of change in Temp/ Humidity in DTH11 Sensor

This, of course, has limitations given by the great variety of types of users that may wish to be included in the IoT. However, the tests carried out suggest that the heart of the prototype would allow providing this, if some facilities at the user interface, equipment and installer's level can be included. This work confirms in relation to the problem of identity scaling and the benefit that can be obtained both with OAuth and with a service-based approach. Delegating authentication to a single point, then relying on temporary credentials, as OAuth2 allows, keeps the identity infrastructure light.

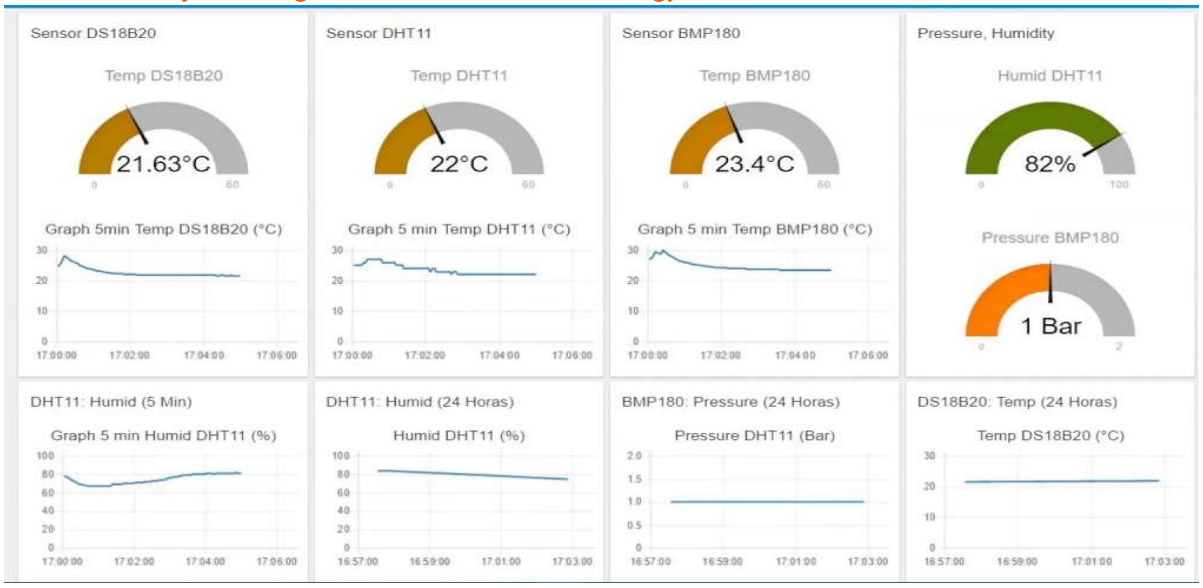

Figure 8: Temp/ Humidity in DTH11 Sensor comparative Analysis

It is precisely this approach to services / microservices that allows working with TLS to not be too demanding at the maintenance level, as highlighted by Díaz-Sánchez et al. This section recognizes the importance that at the security level should be given to the edge equipment; it especially reinforces the gateway equipment in order to later implement improvements in the MQTT protocol, in the style of what was done by Singh et al. but mainly including the use of OAuth2 and TLS.

7. CONCLUSION

This work was motivated by a pressing current need for data management in resource constrained IoT applications, especially those linked to a home context, and to do so without impairing the user's freedom of access to collect information and to modify the configurations of their system. The proposed Framework is centered around the management of data at various levels of the IoT architecture. As discussed at various sections of the paper, the response time of the IoT device is mainly dependent on data generation, resource availability and data management at different levels of the framework. This need, as it turned out, starts among other things from the relative informality of the IoT, especially in a smart home environment. It began with a methodological conception that stratifies the environment in layers: the one most closely related to the interaction with the space by collecting information and executing actions to modify the microenvironment, and the layer of centralized processing and analysis of the information. Both layers were interrelated with a centralized connection that unifies them while also maintaining their light coupling.

This methodology was aimed at allowing an implementation based on microservices, where, in addition to avoiding any type of monolithic structure as much as possible, a group of services and libraries were provided for clients that greatly facilitate the generation of new utilities and components. The tests carried out with the services, clients and sensors generated under this infrastructure confirmed both the robustness and the relative ease of use of the components. The main point, security, although it was not the easiest to implement, once the three defined schemes for the different types of connections were debugged, it proved to be a robust choice, which withstood the tests of improper access. It should be noted, however, that a field-specific methodical testing scheme remains to be designed and implemented, which will be the subject of future work. We can, however, affirm that the combination of KNN, TLS, OAuth2, MQTT, produced the expected results to a large extent.

As main contributions we have the implementation of a data management architecture of IoT for the home; the stable and fluid combination of at least three high-level security management technologies and a functional reference implementation that can be made publicly available for free use.

Acknowledgments

We would like to thank the department of Computer Science Engineering (Lovely professional University) for providing us the support and Lab Facility for our research work.

REFERENCES

*[1] S. N. Swamy and S. R. Kota, "An Empirical Study on System Level Aspects of Internet of Things (IoT)," in IEEE Access, vol. 8, pp. 188082-188134, **2020**. doi: 10.1109/ACCESS.2020.3029847.*

*[2] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum and N. Ghani, "Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations," in IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2702-2733, third quarter **2019**. doi: 10.1109/COMST.2019.2910750.*

*[3] Y. Li et al., "Toward Location-Enabled IoT (LE-IoT): IoT Positioning Techniques, Error Sources, and Error Mitigation," in IEEE Internet of Things Journal, vol. 8, no. 6, pp. 4035-4062, 15 March15, **2021**. doi: 10.1109/JIOT.2020.3019199.*

*[4] F. Meneghello, M. Calore, D. Zucchetto, M. Polese and A. Zanella, "IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices," in IEEE Internet of Things Journal, vol. 6, no. 5, pp. 8182-8201, Oct. **2019**. doi: 10.1109/JIOT.2019.2935189.*

*[5] O. Said, Z. Al-Makhadmeh and A. Tolba, "EMS: An Energy Management Scheme for Green IoT Environments," in IEEE Access, vol. 8, pp. 44983-44998, **2020.** doi: 10.1109/ACCESS.2020.2976641.*

*[6] M. Frustaci, P. Pace, G. Aloi and G. Fortino, "Evaluating Critical Security Issues of the IoT World: Present and Future Challenges," in IEEE Internet of Things Journal, vol. 5, no. 4, pp. 2483-2495, Aug. **2018**. doi: 10.1109/JIOT.2017.2767291.*

*[7] J. Hwang, A. Aziz, N. Sung, A. Ahmad, F. Le Gall and J. Song, "AUTOCON-IoT: Automated and Scalable Online Conformance Testing for IoT Applications," in IEEE Access, vol. 8, pp. 43111-43121, **2020**. doi: 10.1109/ACCESS.2020.2976718.*

*[8] L. Gutiérrez-Madroñal, L. La Blunda, M. F. Wagner and I. Medina-Bulo, "Test Event Generation for a Fall-Detection IoT System," in IEEE Internet of Things Journal, vol. 6, no. 4, pp. 6642-6651, Aug. **2019**. doi: 10.1109/JIOT.2019.2909434.*

[9] O. Said, Y. Albagory, M. Nofal and F. Al Raddady, "IoT-RTP and IoT-RTCP: Adaptive Protocols for Multimedia Transmission over Internet of Things Environments," in IEEE Access, vol. 5, pp. 16757-16773, 2017. doi: 10.1109/ACCESS.2017.2726902.

[10] M. G. Samaila, J. B. F. Sequeiros, T. Simões, M. M. Freire and P. R. M. Inácio, "IoT-HarPSecA: A Framework and Roadmap for Secure Design and Development of Devices and

*Applications in the IoT Space," in IEEE Access, vol. 8, pp. 16462-16494, **2020**. doi: 10.1109/ACCESS.2020.2965925.*

*[11] A. M. Zarca, J. B. Bernabe, A. Skarmeta and J. M. Alcaraz Calero, "Virtual IoT HoneyNets to Mitigate Cyberattacks in SDN/NFV-Enabled IoT Networks," in IEEE Journal on Selected Areas in Communications, vol. 38, no. 6, pp. 1262-1277, June **2020**. doi: 10.1109/JSAC.2020.2986621.*

*[12] J. Bhayo, S. Hameed and S. A. Shah, "An Efficient Counter-Based DDoS Attack Detection Framework Leveraging Software Defined IoT (SD-IoT)," in IEEE Access, vol. 8, pp. 221612-221631, **2020**. doi: 10.1109/ACCESS.2020.3043082.*

*[13] R. Muñoz et al., "Integration of IoT, Transport SDN, and Edge/Cloud Computing for Dynamic Distribution of IoT Analytics and Efficient Use of Network Resources," in Journal of Lightwave Technology, vol. 36, no. 7, pp. 1420-1428, 1 April1, **2018**. doi: 10.1109/JLT.2018.2800660.*

*[14] M. N. Khan, A. Rao and S. Camtepe, "Lightweight Cryptographic Protocols for IoT-Constrained Devices: A Survey," in IEEE Internet of Things Journal, vol. 8, no. 6, pp. 4132-4156, 15 March15, **2021**. doi: 10.1109/JIOT.2020.3026493.*

*[15] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood and A. Anwar, "TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems," in IEEE Access, vol. 8, pp. 165130-165150, **2020**. doi: 10.1109/ACCESS.2020.3022862.*

*[16] M. Yi, X. Xu and L. Xu, "An Intelligent Communication Warning Vulnerability Detection Algorithm Based on IoT Technology," in IEEE Access, vol. 7, pp. 164803-164814, **2019**. doi: 10.1109/ACCESS.2019.2953075.*

*[17] M. W. Condry and C. B. Nelson, "Using Smart Edge IoT Devices for Safer, Rapid Response With Industry IoT Control Operations," in Proceedings of the IEEE, vol. 104, no. 5, pp. 938-946, May **2016**.doi: 10.1109/JPROC.2015.2513672.*

*[18] S. Sathyadevan, K. Achuthan, R. Doss and L. Pan, "Protean Authentication Scheme – A Time-Bound Dynamic KeyGen Authentication Technique for IoT Edge Nodes in Outdoor Deployments," in IEEE Access, vol. 7, pp. 92419-92435, **2019**. doi: 10.1109/ACCESS.2019.2927818.*

*[19] M. Shafiq, Z. Tian, A. K. Bashir, X. Du and M. Guizani, "CorrAUC: A Malicious Bot-IoT Traffic Detection Method in IoT Network Using Machine-Learning Techniques," in IEEE Internet of Things Journal, vol. 8, no. 5, pp. 3242-3254, 1 March1, **2021**. doi: 10.1109/JIOT.2020.3002255.*

*[20] A. A. Simiscuka, T. M. Markande and G. -M. Muntean, "Real-Virtual World Device Synchronization in a Cloud-Enabled Social Virtual Reality IoT Network," in IEEE Access, vol. 7, pp. 106588-106599**, 2019**. doi: 10.1109/ACCESS.2019.2933014.*

*[21] J. An et al., "Toward Global IoT-Enabled Smart Cities Interworking Using Adaptive Semantic Adapter," in IEEE Internet of Things Journal, vol. 6, no. 3, pp. 5753-5765, June **2019**. doi: 10.1109/JIOT.2019.2905275.*

*[22] Y. Cheng, Y. Xu, H. Zhong and Y. Liu, "Leveraging Semisupervised Hierarchical Stacking Temporal Convolutional Network for Anomaly Detection in IoT Communication," in IEEE Internet of Things Journal, vol. 8, no. 1, pp. 144-155, 1 Jan.1, **2021**. doi: 10.1109/JIOT.2020.3000771.*

[23] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal and Q. Z. Sheng, "IoT Middleware: A Survey on Issues and Enabling Technologies," in IEEE Internet of Things Journal, vol. 4, no. 1, pp. 1-20, Feb. ***2017.*** *doi: 10.1109/JIOT.2016.2615180.*

[24] S. Chen, H. Xu, D. Liu, B. Hu and H. Wang, "A Vision of IoT: Applications, Challenges, and Opportunities With China Perspective," in IEEE Internet of Things Journal, vol. 1, no. 4, pp. 349-359, Aug. ***2014****. doi: 10.1109/JIOT.2014.2337336.*

[25] D. Xu and H. Zhu, "Secure Transmission for SWIPT IoT Systems With Full-Duplex IoT Devices," in IEEE Internet of Things Journal, vol. 6, no. 6, pp. 10915-10933, Dec. ***2019****. doi: 10.1109/JIOT.2019.2943377.*