Machine Learning based Access Control Framework for the Internet of Things

Aissam Outchakoucht[1], Anas Abou El Kalam[2], Hamza Es-Samaali[3], Siham Benhadou[4]
LISER Laboratory, Hassan II University, ENSEM School, Casablanca, Morocco, IPI, Paris, France[1, 3, 4]
Cadi Ayyad University, ENSA School, Marrakech, Morocco[2]

***Abstract*—The main challenge facing the Internet of Things (IoT) in general, and IoT security in particular, is that humans have never handled such a huge amount of nodes and quantity of data. Fortunately, it turns out that Machine Learning (ML) systems are very effective in the presence of these two elements. However, can IoT devices support ML techniques? In this paper, we investigated this issue and proposed a twofold contribution: a thorough study of the IoT paradigm and its intersections with ML from a security perspective; then, we actually proposed a holistic ML-based framework for access control, which is the defense head of recent IT systems. In addition to learning techniques, this second pillar was based on the organization and attribute concepts to avoid role explosion problems and applied to a smart city case study to prove its effectiveness.**

***Keywords*—Access control; internet of things; machine learning; security; smart city**

I. Introduction

Access Control (AC) plays a pivotal role in the security world given its mission of protecting digital and physical accesses by delimiting and enforcing who has access to what and in which conditions [1]. However, most of the AC solutions we find in the literature tend to consider the IoT as a single block that is characterized mainly by the limited storage and computing capacities. In this paper, we will come back to this unfair and unrealistic view that slows down the elaboration of a holistic approach to address AC in IoT environments. Moreover, relying on a single technique to address an issue that is as complex as IoT is also a weakness that confines the performance of many IoT security-oriented models.

To fulfil the AC requirements, this paper will progressively build a global framework that not only focuses on policy management and AC models, but also digs deeper into the mechanisms that accurately fit them; which leads to a smooth and coherent Machine Learning (ML) integration going down to highlight what and where ML algorithm(s) should be implemented.

To do so, we first need to delimit the perimeter covered by the IoT by giving a much more representative definition of the term, which will allow us later to tackle the question of AC with a much more appropriate vision, and above all, will lead us to know where and how we can use the power of ML to take advantage of the large amount of objects and data we are handling.

This work was supported by the IGS Group, Paris, France

Motivated by the above, we perceive the relationship between IoT and ML much like the relationship between the human body and its brain. Our bodies gather sensory input such as sight, sound, smell, taste and touch while our brains focus on gathering that data and making sense of it.

The remainder of this paper is presented as follows: Section II exposes an overview of ML applications in IoT scenarios; then Sections III and IV reveals the building blocks of the ML-based framework aiming to handle IoT AC, as well as all the required concepts to understand it. Next, Section V provides the theoretical and technical details of implementation which are applied to a smart city case study, before moving to the last section in which we discuss and evaluate the results.

II. Related Works

Basically, ML algorithms are computer programs that can essentially learn and improve their accuracy by looking at data without being explicitly programmed. In a more formal wording: "A Computer program is said to learn from an experience *E* with respect to some task *T* and some performance measure *P*, if its performance on *T*, as measured by *P*, improves with experience *E*" [2, 3]. In this section, we will be exploring the most promising and latest ML applications used in order to secure IoT environments.

A. Learning Algorithms for Constrained Environments

Despite the common sayings that build a delusive wall between ML algorithms and constrained nodes given the computational and storage limitations of the latter, many studies combined these two worlds in order to answer both application and security issues. In this section we are about to discuss the most recent and relevant ones.

Let us begin with a recent work [4] that combined the strengths of current neural and tree-based learning techniques in conjunction with ternary (-1, 0, 1) quantization to enable computation and size compression of NN models in IoT platforms. This technique outperformed the state-of-the art by 11.1%, 52.2% and 30.6% in the number of computations, the model size, and the overall memory footprint respectively, without losing too much in terms of accuracy.

Another study [5] focused on the IoT device side rather than the cloud. The proposed model is developed on a relatively simple tree learnt in a low-dimensional space for efficient prediction on IoT nodes like Arduino UNO or ATmega328P boards with 16 MHz, 2 KB RAM and 32 KB ROM. The authors executed their model using several datasets

and proved that it was able to make predictions within milliseconds, had lower battery consumption than the state-of-art and could fit in KB of memory.

Additionally, a model called MorphNet was suggested by Google [6] to automate the design of Deep Neural Networks (DNNs). This approach is specifically adjustable to meet constrained environments' requirements without compromising the performance, it actually optimizes the DNN by iteratively shrinking and expanding. The study showed that MorphNet is simple to implement and fast to apply, which is why it is a better choice for IoT scenarios.

ShuffleNet [7] is another contribution in this direction. It is a particularly computation-efficient CNN designed specifically for mobile devices which are essentially characterized by their limitations in terms of computational power. It is mainly based on the power of 1×1 convolutions combined with channel shuffle with the aim of reducing the required cost for computation without neglecting the accuracy. Being implemented on an ARM-based mobile device, the model attains up to 13× actual speedup over AlexNet.

Besides, the authors in [8] suggested a NN-based implementation that takes benefit from the communications passed between IoT nodes. Theoretically, this work is founded on the UAT theorem affirming that a NN with a single hidden layer is enough to compute a bounded approximation of a generic continuous function. In fact, this remark has led to integrate intelligence into IoT constrained platforms by means of some (local and on-the-fly) computations as the data navigate between the IoT devices using the collective behavior of such networks.

A Mobile and Edge Computing (M/EC) solution was proposed in [9] to bring computation near the IoT end-nodes by applying CNNs, RNNs and RL at the edge of IoT networks. The very idea of this work is to implement Information-Centric Networking on top of the IoT via some techniques namely shared weights, pooling, and in-network caching to solve storage issues on IoT nodes. This approach led to remarkable reduction in latency for time-critical applications.

Another study [10] digs deeper into the technical hardware requirements to implement DL algorithms over IoT devices. The authors implemented several models inside three boards: Qualcomm Snapdragon 800 used for phones and tablets (4-core 2.3 GHz CPU, 1GB of RAM and 8MB DSP), Intel Edison principally oriented to wearables and form-factor sensitive IoT (500MHz dual-core CPU, 1 GB of RAM) and finally Nvidia Tegra K1 used for example in June IoT Oven [11], Nexus 9 phone and IoT-enabled cars (up to 1.7GB of RAM). The study proved, inter alia, the feasibility of implementing DL techniques on IoT oriented boards.

A more general approach was proposed in [12], in which the authors came with a semi-supervised deep RL model designed to fit smart cities. Its inference engine exploits Variational Auto Encoders (VAE) to generalize optimal policies. The model was implemented to handle localization issues in a smart building case study portrayed as an ensemble of labeled positions associated with the Received Signal Strength Indicator values from multiple iBeacons. It was able to learn better action policies with at least 23% improvement in terms of distance to the target as well as almost 67% more gathered rewards compared to the supervised Deep RL approaches.

B. Learning Applications for IoT Security

Now that we have seen many ML-based applications in the IoT, let us move to some studies that tackled security problems (always in IoT environments) through ML tools.

In fact, Support Vector Machines (SVMs) are one of the first and most used ML models. They represent standard classification models, generally known for splitting hyperplanes. Data sorting is achieved through maximizing the distance between the hyperplane and the nearby training samples of each class. SVMs are more adapted to datasets with a large number of features but a relatively small number of samples [13]. In the IoT world, a study [14] proposed a linear SVM-based Android malware detection system to secure IoT platforms. The comparison that was led between the performance of the model and other ML algorithms outbalanced the SVM method. Besides, SVM was also used to compromise cryptographic devices [15, 16]. However, one of the big challenges in multidimensional SVM problems is the tough task of selecting a suited kernel.

Another generic method is Random Forest (RF): an accumulation of Decision Trees (DTs), which means that they are built and trained in order to vote for the output class [17]. A study [18] over 17 IoT devices belonging to nine categories affirmed that RF (among other ML methods) presents significant improvements in the identification of unauthorized IoT nodes. Another ML-based study [19] was performed on IoT environments to detect DDOS attacks. In this regard, RF provided slightly superior results compared to other ML methods. That being said, it is important to emphasize that RF methods are not always feasible, specifically over large datasets as they require the construction of a -relatively- large number of DTs.

In another direction, UL is represented by the popular K-Means with the key objective of Data clustering (k being the number of clusters). The algorithm consists of assigning each data sample to one of the k clusters based on their (similar) features. Usually UL models are privileged when the dataset is not labelled. In IoT, k-means clustering was used to distinguish Sybil attackers from normal sensors through clustering the channel vectors in industrial WSNs [20]. Nevertheless, this technique has many limitations, namely the need to have roughly equal numbers in each cluster for the algorithm to properly work, as well as the non-trivial task of choosing k [21].

Now, let us move to the deep sphere, and begin our survey by Convolutional Neural Networks (CNNs). Actually, the basic idea of a CNN is to put a bit of structure in NNs [22] by shrinking the enormous number of connections between layers, thus optimizing the training time. One of the main benefits of CNNs is their end-to-end nature ensured by their built-in "features extraction" ability. Yet CNNs still have a high computational cost; hence the difficulty of implementing

them on IoT constrained nodes. Many studies managed to bypass this limitation though, especially using distributed architectures [23]. Another study showed that CNNs could help in Android malware detection by means of raw sequence static analysis (RSSA) of disassembled programs [24].

One more giant pillar of ML nowadays goes under the name of Recurrent Neural Network (RNN). It is undoubtedly one of the ML big discoveries thanks to their particularity of having memory. They can read inputs $\boldsymbol{X}^{<t>}$ in sequence, and "remember" some information/context thanks to their hidden layer activations that get passed from a given time-step to the following [22]. Accordingly, RNNs can achieve excellent results in classifying network traffic and detecting malicious behavior. Besides, RNNs could be a good choice for IoT since it produces massive sequential data from different nodes. For instance, a previous work [25] proved the worth of using RNNs to detect network traffic behavior by modeling it as a sequence of states changing over time. Yet, vanishing and exploding gradient problems still the ultimate nightmare of RNNs.

In another direction, many researchers consider that there was various contributions in ML in recent times, but Generative Adversarial Networks (GANs) are the only contribution that could be called a breakthrough in the last decade. GAN trains two models at the same time: a generative model G to identify the data distribution, and a discriminative model D to predict the probability that a sample came from the training data rather than G [26]. A recent work [27] realized a GAN-based architecture in order to secure IoT systems by detecting abnormal behavior. GANs may have a potential application in IoT security especially in zero-day-like threats given their ability to learn diverse attack scenarios and then to generate innovative attacks beyond the existing ones. Though, up to now the training phase of GANs still unstable and a tough task [21].

Providing a large amount of training data is not always an easy task; hence, finding alternatives is a matter of serious concern for ML experts. Reinforcement Learning (RL) consists of learning behavior only through interactions between an agent (usually represented by the algorithm) and its surrounding environment; in fact this learning process consists of increasing the rewards it receives from the environment. Many researchers focus on the application of RL to IoT security; for instance, the work in [28] opted for an RL approach to learn a sub-band selection policy so that it could avoid both jammer signals as well as interference from other radios in wideband autonomous cognitive radios (WACRs). Two of our previous works [1, 29] tackled the Access Control (AC) in IoT scenarios, the two building blocks were: first taking into account the smart devices' context while making an AC decision; and proposing AC policies that can be improved and optimized over time. However, given the enormous and heterogeneous amount of data generated by IoT devices, the proposition benefits from the power of RL, to accomplish this task. The problem with RL algorithms is that they require a large number of practice run (given their trial-error nature) before they can make significant progress.

It is worth noting that, in addition to provide an explicit survey of the latest and relevant works in IoT and ML, one of the motivations of this related works section is to prove that ML is already used in the IoT world, and consequently to disprove the idea claiming that IoT and ML are two parallel universes. In the following section, the proposition of this paper is presented with all the necessary details.

Table I summarizes and compares these studies especially based on their achievements in a number of IoT situations.

TABLE. I. COMPARISON AND SUMMARY OF ML STUDIES FOR IOT

Application domains	Used algorithms	Achievements	Studies
Networks traffic optimization in indoor or smart city scenarios	Semi-supervised D-RL Auto Encoders D-NN, CNN, RNN	Bring computing next to IoT devices Reduction in latency More gathered rewards	[9], [10], [12]
(Relatively) Simple processing situations	D-NN, CNN Low-dimensional trees	Local & on-the-fly computations Up to 13× actual speedup over AlexNet Decreasing model size, memory consumption Prediction within milliseconds	[4], [5], [6], [7], [8]
Malware & Intrusion Detection	SVM, RF, CNN	Significant improvements in the identification of unauthorized IoT nodes Compromising cryptographic devices	[14], [15], [16], [18], [19], [23], [24],
Network threats & Network traffic behavior	RNN, GAN	Zero-day attacks Good results in time-based environments Data augmentation	[25], [27]
Security policy improvements	RL	Policy optimization Policy efficiency	[1], [28], [29]
New and unprecedented attacks	K-means, RL	Zero-day & Sybil attacks detection Avoid jammer signals	[1], [20], [28], [29]

III. PRELIMINARIES

We begin this section by exposing the research questions standing behind our work, together with all the essential details required to understand our proposition.

A. Problematic

The nature of a large portion of IoT devices (e.g. Healthcare, critical infrastructures) makes security the number one priority: it is, literally, a matter of life and death. In addition, the density, heterogeneity and autonomy are intrinsic characteristics of these systems that not only expend the perimeter of potential attacks but also their magnitude [21].

However, it turns out that treating all security aspects in one proposition is not a reasonable task; hence, this paper is focusing on the AC cornerstone because of its nature of protecting the access to digital and physical resources by delimiting, managing and enforcing who has (has not, is obliged to have) access to what, when and under which conditions [1].

Furthermore, the abovementioned IoT features impose intelligent and dynamic management instead of traditional and impractical one; we believe that IoT must benefit from these features considered so far as obstacles, IoT nodes need to "learn to look after each other". To do so, our research has confronted many speed bumps that we have tried to demolish in this article.

First, the misleading idea that reduces the IoT to constrained devices, but more importantly: The need to exploit the quantity of IoT devices and data as a catalyst for security to emerge.

Not to mention the necessity for models that go beyond simply defining AC policies to understand the context of each smart device and continuously improving these policies, without falling into the trap of static management or role explosion.

To the best of our knowledge, there seems to exist no previous work presenting a holistic ML-based framework for IoT answering these problematics.

B. IoT and Computation Paradigm

At first sight, IoT is a concatenation of two words: "Internet", which refers to connectivity and communication aspects; and "Things", which is a generic and global term that includes all kinds of objects, whether large or small, powerful or not. In that sense, every "thing" endowed with communication capacity is an IoT device. Put this way, one can easily classify the aforementioned constrained nodes, together with mobile phones, a Raspberry Pi board and cloud servers as IoT devices; and can also set traditional TVs, calculators or pillows outside the IoT scope (unless they are connected).

Actually, no one can deny the difficulty (sometimes even the impossibility) of implementing complex ML tools on several types of constrained nodes. Yet, these latters remain ambiguous especially given the recent innovations in ML-oriented chips, which are mainly due to the excessive demand and hot market of AI applications these days. This subsection exposes three reasons to motivate researchers –and investors– not to draw a spontaneous correlation between IoT and ML ineptness:

- IoT > constrained nodes: As explained before, one key idea to clarify when talking about IoT is that it is more than just a collection of constrained nodes. Not to confound with Wireless Sensor Networks.
- Hardware progress: The AI market is in an exponential growth, which leads to more investments, then to more innovations. This climate could only be beneficial for ML community. With this in mind, one can take a look at Amazon store for example to see how the ratio of hardware size to its storage capacity is decreasing faster than ever before. Regarding computation capacities the progress is astonishing as well, for instance, just few months ago, NVIDIA announced a 70mm x 45mm AI computer, 4 GB memory, Quad-core ARM® A57 CPU and 128-core NVIDIA Maxwel GPU [30].
- Software evolution: What is true for hardware also holds for software. Section II is an illustration of the active race in proposing new and suitable ML algorithms for IoT. Besides, many dedicated and extremely optimized ML libraries are already easing programmers' life. For instance, Google's Tensorflow Lite transforms heavy TensorFlow models into compressed flat buffers, which are then loaded into a mobile or embedded device [31].

Furthermore, there are several active research directions that could lead to further findings (even breakthroughs) in IoT-adapted ML applications: (i) parallel computing in training phase using Graphical Processing Units and Tensor Processing Units (GPUs/ TPUs), (ii) transfer learning in order to swiftly transfer the knowledge from pre-trained models, (iii) fog computing to decrease communications overhead size, data traffic, user-side latency, (iv) fast optimization algorithms [32].

C. Background

The aim of this section is to explore two concepts that are essential to understand our proposition, namely, AC and IoT architecture.

In fact, AC is of paramount importance being the entry point of every system after the identification/ authentication phase, thus securing any system must pass through (if not begin with) controlling its accesses. In the literature, tens of models are handling this issue, one of the most popular is Role Based Access Control [33] (RBAC); without going into too much details: instead of granting (or removing) a separate permission to every subject in the network, the model aggregates these subjects by roles and thus gets a lightweight version of its AC policy. Yet, end devices are not involved in AC decision, also, even with the aggregation of subjects into

roles, IoT systems still have astronomical number of objects and actions to manage. Attribute Based Access Control [34] (ABAC) is another model that is taking up more and more space recently, its basic concept is to identify subjects and objects through attributes (characteristics), then the permissions are granted according to these attributes, which could be any relevant security-characteristics, this makes ABAC, unlike RBAC, more adapted to afford fine-grained AC highly valued in IoT circumstances. Still, it comes with a terrifying and intolerable drawback: complexity. Other models [35] tried to make many tradeoffs but generally fall into one of the aforesaid limitations.

Reasonably, Organization Based Access Control [36] (OrBAC) is a remarkable model that we believe it could be used as a foundation to an IoT-oriented solution. It inherits and extends the benefits of RBAC by proposing abstractions to all the AC elements (subject to role, object to view and action to activity – see Fig. 1), then it adds two more dimensions to the decision making process: context (crucial for IoT) and organization; this latter makes it a centralized model, but we will see in the next section how to turn this limitation into a strength.

In the other hand, and given what is been elucidated earlier in this paper, IoT is not a single homogenous block; hence, in order to propose reasonable solutions, there is no other way but to have a conception of the main building blocks composing standard IoT environments. To do so, of course one could suggest several subdivisions but since the intention of this paper is to conciliate IoT and ML, the proposed categorization needs be centered on computing capabilities.

In that sense, many researchers [37, 38] agree that every IoT platform could be molded into one or more of the following categories: C1: the constrained layer, here is where the constrained devices are located (physical constraints on many characteristics such as size, weight, available power and energy [39] which make them unable to accomplish more than basic tasks); C2: this category includes more powerful nodes, which are capable of executing relatively serious computations. In fact, the vast majority of the everyday smart devices fit into this category (e.g. smartphones, smart homes components); finally, C3: Computational or offloading layer, it is the most powerful one, it could be Cloud or local servers, computers, GPUs/TPUs and so on.

Generally, complex environments (like smart cities) are a combination of the three layers. Another point to underline is the absence of explicit and rigid boundaries between these layers; this is mainly due to the dynamic nature of the IT domain, in fact what we call powerful today (or for a given task) might be considered constrained tomorrow (or for another task) and vice versa. Based on some examples, Fig. 2 exposes the borders and intersections between these categories.

As shown above, typically what most people call IoT is in general the intersection between the three layers, it is where all sorts of devices are interacting with each other to create this large universe of smart devices.

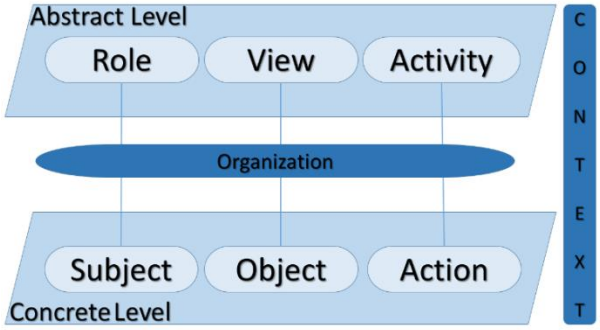

Fig. 1. Simplified Presentation of OrBAC Layers.

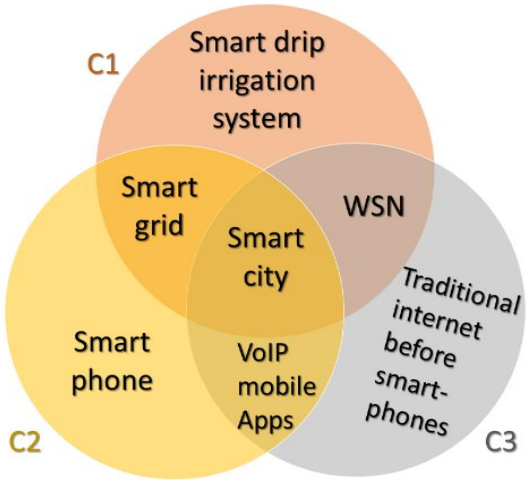

Fig. 2. Intersections between IoT Layers.

IV. Contribution

In this section we present the AC framework that takes into consideration all the previously discussed requirements.

A. Global Questions Need Global Answers

One can fairly claim that IoT is the largest and most heterogeneous artificial network humans have ever made, it is becoming earth's nervous system. Therefore it would be naive, if not irrational, to search for elementary narrowed solutions to such a complex and multifaceted problem; instead what this paper is suggesting is a multi-layer AC solution, that exploits ML and OrBAC strengths to answer IoT burning questions exposed in Section III-A.

First thing to consider when treating AC in IoT is the overall architecture, whether it is distributed (like blockchain based solutions [40]) or centralized as long as it is equipped with the collaboration aspect. In fact, each of these two architectures has its advantages and drawbacks [29]; however, without loss of generality, this paper mainly focus on the second one for the following reasons: (i) IoT, at least as we know it today, requires that each device has an owner, whether a person, in the case of a smart home for example or an organization in industry or in smart cities. So practically an approach that takes this aspect into consideration will be closer to reality. (ii) Even in distributed architectures, there always should be an entity in charge of defining AC policies for the IoT nodes, unless the device is open to everyone (and

therefore no need for AC at all). For simplicity reasons, in both scenarios this entity is given the name "object owner" (*OO*).

As shown in Fig. 3, the object owner *i* (OO_i) owns several IoT devices which can belong to any category (C1, C2 or C3). He defines their AC policies and stores them either locally, in a distributed manner on multiple servers or even in large networks (like blockchain) using smart contracts. Henceforth, the location of these policies is called Policy Information Point (PIP), as defined by the ISO/IEC standard for the access control framework [41] and the XACML related architecture [42].

As mentioned before, the choice of OrBAC as the background model to an IoT-oriented framework is defended by the following reasons: First, it has the concept of organization by design thus no need for extra dimensions to designate the *OO*. Also, OrBAC is distinguished by two other features crucial for IoT environments, namely an advanced level of abstraction required to alleviate the complexity produced by the colossal number of devices; together with the context incarnated in all OrBAC rules, which will ease the collect of real time contextual information from the end nodes for better AC decisions. In a more formal sense, the AC policy is stored in the PIP as rules presented in this form:

$$permission(org, r, v, ay, c) \quad (1)$$

Where *org* stands for organization or the view owner, *r* for role (aggregation of subjects), *v* for view as a collection of objects, *ay* for activity which is an abstraction of actions whereas *c* is the context. Thus the previous rule declare that: In the organization *org* the role *r* has permission to execute the activity *ay* on the view *v* under *c* circumstances.

1) Pre-request stage: Policy initiation: In step one, when the *OO* have to define a new rule, either the device *o* fits in one of the existing views *v* so the affectation : $use(org, o, v)$ (2) is executed; if not the role is automatically created when declaring a new permission and the subject and role get the same name.

Another key concept of this framework is the process of matching abstract and concrete entities, it also begins in phase one: In fact, besides the aforementioned rules, the PIP contains two types of match functions:

Usual correspondences in the form of $match(org, s_j, r_i)$ as shown in (2) are employed for the frequently used entities, while generic match functions based on attributes, for instance:

$$match(org, s^1, \dots, s^n, r_i) \quad (3)$$

Which means if a subject *s*, has the attributes $s^1, \dots, s^n$ then it belongs to the role r_i. Yet since the designation of s_j could as well be considered as an attribute of itself, we can use (3) for both.

2) Inference stage request processing: Now that AC policies are initiated and stored in the PIP, phase 2 begins: a subject s_i is willing to execute an action a_i on an object o_i, to do so the following request is sent to the Policy Enforcement Point (PEP):

$$get\_access(org, s_i, o_i, a_i, c_i) \quad (4)$$

When the PEP gets the access request, it triggers the process of matching (phase 3); it is the step where we go up from concrete to abstract entities in order to reduce the complexity. So the PEP transfers (4) to the Policy Decision Point (PDP), which in its turn requests the PIP by an:

$$get\_match(org, s_i, o_i, a_i, c_i) \quad (5)$$

After that, the PIP, responds by the corresponding matches:

$$back\_match(org, r_i, v_i, ay_i, C_i) \quad (6)$$

Up till now, the PDP has all the static features to take the decision. However, even if we have already handled two IoT major worries, namely context and complexity, the process still lacking dynamism. In fact, policies are statically stored in the PIP without any learning from past experiences. To answer this, (6*) will be coupled with an extra feature: a ratio reflecting the probability of a safe access granting given the aforementioned characteristics:

$$back\_match(org, r_i, v_i, ay_i, C_i, p) \quad (7)$$

To do so, (1*) needs to be joined with the probability feature which is set by default (in the policy definition phase) to *p=1*, then it is updated over time:

$$permission(org, r, v, ay, c, p) \quad (8)$$

Now, and based on a threshold fixed by the *OO*, the PDP can decide (phase 4) and inform the PEP, and consequently the requester, about the final decision. Note that this threshold has two essential benefits: first it gives the organization a better personalization of the framework, in addition to allowing it to define even several thresholds given the criticality of certain resources or context.

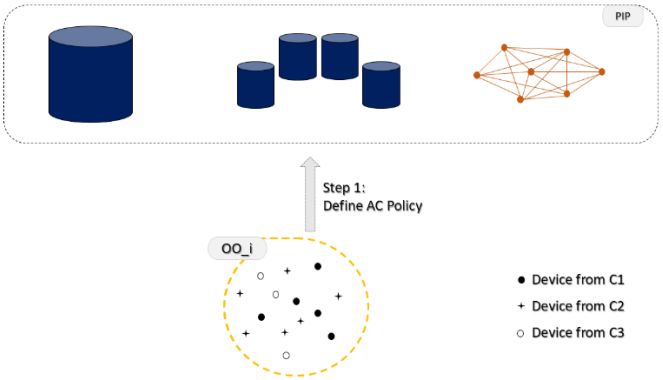

Fig. 3. Step 1: to define AC Policies.

3) Post-request stage learning and upgrading: The final phase -5- in our process is a post access one, it consists of calculating a feedback rating of the experience, which will be used by the PDP to generate AC policy updates leading to more accurate decision in the future (a concrete example is given in the case study section). The output will be stored in the "*learning matrix*" which will have 6 columns (organization, role, view, activity, context, feedback) and as many line as the number of experiences the system could store; while feedback is a rational number between 0 and 1.

The learning algorithms run in phase 5 varies according to the hardware resources of the system but also according to the layers defined in Section III-C. However, generally in complex and multifaceted IoT environments we propose using: RL and many resource consuming SL scheduled, for example, periodically in category C3; SL up to a reasonable size of the learning matrix in C2; while leaving normal equation stechnique or no ML at all to C1. Table II summarizes the previously detailed steps.

For simplicity reasons, we were focusing in permission formulas, however what goes for permissions is also valid for obligations and prohibitions [43].

B. The Algorithm

The steps discussed in Table II are compressed in the following Fig. 4 to explain the overall functioning of the algorithm. In fact, the framework could be segmented into three main time frames: (i) pre-request tasks, which handle the definition of the AC policies; (ii) request processing, involving all the actions triggered after an access request up till the subject receives back a permission/rejection; (iii) post-request actions that are responsible for the learning and policy improvements.

Note that the proposed framework is a decentralized one. In fact, the concept of organization is introduced to decompose complex IoT environments into reasonable and manageable groups, not to turn them into one giant centralized one. For instance, if we have to manage AC in a smart city situation, the framework will treat this multidimensional platform as a collection of organizations interacting and collaborating with each other.

Several studies have examined the collaboration issues in OrBAC [37, 44, 45, 46], either by creating further abstract entities, web services, or even through prior agreements between the involved organizations, however the definition of AC relationships using attributes that we saw in 1.b. (Table II) is a better alternative in IoT situations since with one tool we answer both intra and inter organizations AC concerns. An example of this scenario will be discussed in the following section, where we are exploring a smart city case study.

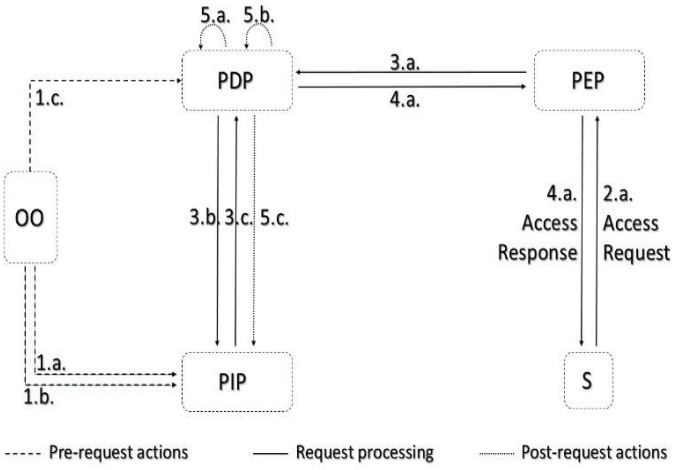

Fig. 4. Overall Functioning of the Framework.

TABLE. II. SUMMARY OF THE FRAMEWORK'S PHASES

	Action	Responsible	Location/ destination	Example
Phase 1	**1.a.** AC policy definition	OO	PIP	$\text{permission}(\text{org}, \text{r}, \text{v}, \text{ay}, \text{C}, \text{p})$
	1.b. AC relationship definition (either explicit or through attributes)	OO	PIP	$\text{empower}(\text{org}, \text{s}^1, \dots, \text{s}^m, \text{r})$ $\text{use}(\text{org}, \text{o}^1, \dots, \text{o}^n, \text{v})$ $\text{consider}(\text{org}, \text{a}^1, \dots, \text{a}^p, \text{ay})$
	1.c. Threshold definition	OO	PDP	$tr = 1; \quad tr = 0.7$
Phase 2	**2.a.** The subject request executing an action over a resource/object	The requester/ subject	PEP	$\text{get\_access}(\text{org}, \text{s}, \text{o}, \text{a}, \text{c})$
Phase 3	**3.a.** Request for decision	PEP	PDP	$\text{get\_access}(\text{org}, \text{s}, \text{o}, \text{a}, \text{c})$
	3.b. Request matching information	PDP	PIP	$\text{get\_match}(\text{org}, \text{s}, \text{o}, \text{a}, \text{c})$
	3.c. Matching response	PIP	PDP	$\text{back\_match}(\text{org}, \text{r}, \text{v}, \text{ay}, \text{C}, \text{p})$
Phase 4	**4.a.** Make decision	PDP	PEP	$\text{grant\_access}(\text{org}, \text{s}, \text{o}, \text{a}, \text{c})$ $\text{deny\_access}(\text{org}, \text{s}, \text{o}, \text{a}, \text{c})$
Phase 5	**5.a.** Update learning matrix	PDP	PDP	$\text{l\_matrix.append}([\text{org}, \text{r}, \text{v}, \text{ay}, \text{C}, \text{p}])$
	5.b. Learning process	PDP	PDP	$\text{model.fit}(\text{l\_matrix})$
	5.c. Send updates periodically	PDP	PIP	$\text{permission}(\text{org}, \text{r}, \text{v}, \text{ay}, \text{C}, \text{p})$

V. Case Study

The IoT is growing by leaps and bounds, thus creating this large, smart and autonomous system requiring less and less human intervention. Smart city (SC) was always the case in point that portrays this vision where devices not only interact but depend on (even control) each other. This section presents a SC situation to depict how the previously described framework could be implemented in such complex environment.

Actually, the choice of SC has been motivated by many reasons: First, because it covers many of the other IoT use cases. Secondly, in addition to the IoT world, it is also a typical case in the AI domain. Thirdly, it presents an active research field [47], and last but not least it is becoming a necessity in order to efficiently deal with the massive growth of urbanization that is estimated to reach 66% by 2050 [48].

To better illustrate this example, let us take three organizations in a SC, namely: a car rental agency (CRA), which represents the organization we are willing to secure, a smart parking (SP) and a police station (PoS).

As a CRA, our organization is mainly composed of self-driving cars (decomposed into two views: luxury and normal cars); its customers are generally normal clients, VIP clients or blacklisted ones (which makes respectively three roles: NC, VIP, BC); regarding the activities it is more realistic to categorize them by rental period (a1: 1 day, a2: between 1 and 3 days, and a3: more than 3 days). Finally, the context represents the time of the year during which the request is made: is it a peak season (peak) like summer for example, or off season (off).

In this case, each self-driving car is equipped with its own PEP, which means it is the car itself that receives the access request and responds the requester by the final decision. PIP may ideally be one (or several) local server(s) storing the AC policy. Regarding the PDP, in general it needs to be spread out over two layers: one part dealing with the steps (1.c., 3, 4 and 5.a.) that are executed within the smart car, and another part that needs to be run on C3 category (namely 5.b. and 5.c.). Nevertheless, as we will see later some lightweight versions of these two steps could also be run within the smart car. The following Fig. 5 portrays the building blocks forming this platform:

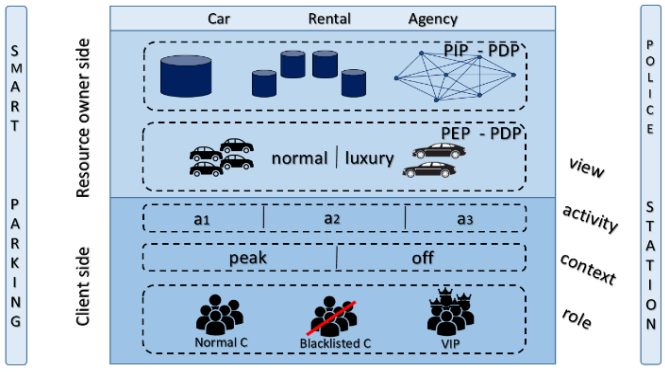

Fig. 5. The Building Blocks of the Case Study.

Now that we have exposed all the stakeholders, first thing to do as the owner of CRA is to define a primary AC policy, so in the PIP there will be two sorts of rules: (i) Those of the 1.a. step, which will be in the form of:

$$permission(*, VIP, luxury, a3, peak, 1)$$

$$prohibition(Competitor\_CRA, BC, normal, a1, off, 1)$$

* Star stands for *all*, i.e. any organization.

Then the definition of the AC relationships, which are responsible for matching abstract to concrete entities, for example a car is considered luxury if its release date comes after 2017 and its price exceeds 100,000$:

$$consider(*, time \geqslant 3days, a3)$$

$$empower(*, badge = True, VIP)$$

$$use(CRA, r\_date \geqslant 2017, price \geqslant 100{,}000\$, luxury)$$

For luxury cars the threshold is set to 1, so that access is not granted to any role without a 100% confidence about its previous experiences. While for the less critical view (*normal*) the threshold is reduced to 0.8 allowing more usability. These thresholds are set in the car side's PDP.

Let's move to phase 2. At the moment, an access request arrives to the PEP of the car id=79, it is a subject presenting his membership badge and demanding this resource for 5 days starting from August the 1st, 2019.

$$get\_access(org_A, badge = True, id = 79, time = 5days, starting\_time = 01/08/2019)$$

After the previous request is forwarded to the PDP, this latter requests matching information from the PIP via:

$$get\_match(org_A, badge = True, id = 79, time = 5days, starting\_time = 01/08/2019)$$

Then it gets the response:

$$permission(org_A, VIP, luxury, a3, peak, 1)$$

And since the threshold for luxury cars is set to 1, the PDP makes and informs the PEP about its final decision allowing the requester to access/use the car:

$$grant\_access(org_A, badge = True, id = 79, time = 5days, starting\_time = 01/08/2019)$$

Now comes the learning phase. Actually, it is up to the organization either to rate the experience after its end (i.e. rate just the physical state of the car for example) or to perform an online rating so that it gets periodic feedback from the smart car to compute more sophisticated inferences (e.g. geolocation, fuel consumption, speed) and then the PDP ends up with a weighted average. To simplify our case, we use the first option.

Therefore, after the car is back, the PDP receives the required ingredients to compute the feedback. Let us consider

that what is important for this CRA is whether the car is back in time and its mechanical situation, and that for both features this experience was negative, so the feedback was set to 0.6. The learning matrix is now updated:

$$l\_matrix.append([org_A, VIP, luxury, a3, peak, 0.6])$$

Now we can imagine that if after a while similar feedbacks comes from the same organization (org_A) or under the same context or whatever, the algorithm will detect the pattern when the learning model is run over the collected data using $model.fit(l\_matrix)$, then the AC policy could eventually be updated, thus improved over time.

A final point that we want to highlight is about the attributes used in the matching phase. In fact, depending on their nature there are two ways to collect them; either explicitly as we saw in the previous example (i.e. as parameters within the access request) or by extracting them directly from the object. For instance, if we want to use the license plate number we could eventually use the cameras from the smart parking.

VI. Discussion and Conclusions

The motivation behind this work was to come up with a smart, decentralized and IoT-suited AC framework. In this section we discuss theoretical and practical contributions of this paper as well as their strengths over existing solutions.

First thing to remember is that IoT are not all constrained. To prove this we went back to define this paradigm and to delimit its boundaries and layers; in addition to overview several IoT propositions that have used ML techniques. However, to the best of our knowledge, the existing solutions are narrowed ones, each one focus either on proposing a model, managing the policy or tackling specific techniques. The problem with this approach shows up when an organization wants to put it all together, generally the concatenation of these uncoordinated solutions do not give acceptable results. Thus proposing a holistic framework to manage AC in IoT environments is another key contribution of this paper.

Equally important, the introduction of the notion of organization in IoT is notably benefic since it helps drastically decreasing the problem of role explosion, which is the number one problem challenging role-based and attribute-based AC solutions. In fact, an IoT environment could always be broken off into several organizations and therefore, depending on their mission, the roles will be manageable; and of course what goes for roles also goes for views, activities and context. Note that the organization aspect has by no means been a source of centralization, it is rather a push toward more decentralization and collaboration.

Another interesting and captivating point: the learning process we have introduced actually differs from the traditional procedure commonly used in the current AI applications, which consist first of a learning phase followed by the prediction phase. In our framework it is rather a minimum of basic knowledge is initiated in the beginning then learning came to fine-tune this expertise. It is actually how humans learn, they always have some innate skills before anyone comes to teach them anything.

Finally, we believe that having the ability to personalize the threshold, not forcing an immediate update after each experience, and allowing explicit as well as extracted attributes bring this framework with further flexibility and adaptability to fit the IoT requirements discussed in SectionIII-A.

References

[1] A. A. El Kalam, A. Outchakoucht, H. Es-samaali, “Emergence-Based Access Control: New Approach to Secure the Internet of Things”. DTUC ’18 Paris. 2018.

[2] T. M. Mitchell, “Machine learning”, 7th ed. NY, McGraw Hill, ISBN: 0070428077. 1997.

[3] Y. LeCun, Y. Bengio, G. Hinton, “Deep Learning”, Nature, volume 521, 2015.

[4] D. Gope, G. Dasika and M. Mattina, “Ternary Hybrid Neural-Tree Networks for Highly Constrained IoT Applications,” in the 2nd Conference on Systems and Machine Learning (SysML), 2019.

[5] A. Kumar, S. Goyal, M. Varma, “Resource-efficient Machine Learning in 2 KB RAM for the Internet of Things”; Proceedings of the 34th International Conference on Machine Learning, 2017.

[6] A. Gordon et al. “MorphNet: Fast & Simple Resource-Constrained Structure Learning of Deep Networks,” arXiv:1711.06798. 2017.

[7] X. Zhang, X. Zhou, M. Lin and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6848-6856, 2018.

[8] N. Kaminski et al., "A neural-network-based realization of in-network computation for the Internet of Things," IEEE International Conference on Communications (ICC), pp. 1-6, 2017.

[9] H. Khelifi et al., "Bringing Deep Learning at the Edge of Information-Centric Internet of Things," in IEEE Communications Letters, vol. 23, no. 1, pp. 52-55, 2019.

[10] N. D. Lane et al., "An Early Resource Characterization of Deep Learning on Wearables, Smartphones and Internet-of-Things Devices," in the Proceedings of the 2015 International Workshop on Internet of Things towards Applications, pp. 7-12, 2015.

[11] Nvidia. [online] Available at: https://blogs.nvidia.com/blog/2015/06/09/gpu-powered-june-oven/, [Accessed 2 December 2019].

[12] M. Mohammadi, A. Al-Fuqaha, M. Guizani and J. Oh, "Semisupervised Deep Reinforcement Learning in Support of IoT and Smart City Services," in IEEE Internet of Things Journal, vol. 5, no. 2, pp. 624-635, 2018.

[13] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153-1176, 2015.

[14] H.-S. Ham, H.-H. Kim, M.-S. Kim, and M.-J. Choi, "Linear SVM-based android malware detection for reliable IoT services," Journal of Applied Mathematics, vol. 2014, 2014.

[15] A. Heuser and M. Zohner, "Intelligent machine homicide," in International Workshop on Constructive Side-Channel Analysis and Secure Design, pp. 249-264, 2012.

[16] L. Lerman, G. Bontempi, and O. Markowitch, "A machine learning approach against a masked AES," Journal of Cryptographic Engineering, vol. 5, no. 2, pp. 123-139, 2015.

[17] L. Breiman, "Random forests," Machine learning, vol. 45, no. 1, pp. 5-32, 2001

[18] Y. Meidan et al., "Detection of Unauthorized IoT Devices Using Machine Learning Techniques," arXiv preprint arXiv:1709.04647, 2017.

[19] R. Doshi, N. Apthorpe, and N. Feamster, "Machine Learning DDoS Detection for Consumer Internet of Things Devices," arXiv preprint arXiv:1804.04159, 2018.

[20] Q. Li, K. Zhang, M. Cheffena, and X. Shen, "Channel-based Sybil Detection in Industrial Wireless Sensor Networks: a Multi-kernel Approach," in IEEE Global Communications Conference, pp. 1-6: IEEE, 2017.

[21] M. Al-garadi, A. Mohamed, A. Al-ali, et al. "A survey of machine and deep learning methods for internet of things (IoT) security". arXiv preprint arXiv:1807.11023, 2018.

[22] M. Ford, "Architects of Intelligence: The truth about AI from the people building it", book, Packt Publishing, 2018.

[23] E. De Coninck et al., "Distributed neural networks for Internet of Things: the Big-Little approach," in International Internet of Things Summit, pp. 484-492, 2015.

[24] N. McLaughlin et al., "Deep android malware detection," in Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, pp. 301-308, 2017.

[25] P. Torres, C. Catania, S. Garcia, and C. G. Garino, "An analysis of Recurrent Neural Networks for Botnet detection behavior," in Biennial Congress of Argentina, IEEE pp. 1-6, 2016.

[26] I. Goodfellow et al., "Generative adversarial nets," in Advances in neural information processing systems, pp. 2672-2680, 2014.

[27] R. E. Hiromoto, M. Haney, and A. Vakanski, "A secure architecture for IoT with supply chain risk management," in 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), vol. 1, pp. 431-435, 2017.

[28] M. A. Aref, S. K. Jayaweera, and S. Machuzak, "Multi-agent Reinforcement Learning Based Cognitive Anti-jamming," in Wireless Communications and Networking Conference (WCNC), IEEE pp. 1-6, 2017.

[29] A Outchakoucht, ES Hamza, JP Leroy, "Dynamic access control policy based on blockchain and machine learning for the internet of things," in International Journal of Advanced Computer Science and Applications, Vol. 8, No.7, 2017.

[30] NVIDIA Corporation, Jetson Nano Developer Kit, March 2019, [online] Available at: https://developer.nvidia.com/embedded/jetson-nano-developer-kit [accessed 2 December 2019]

[31] Google Brain, Deploy machine learning models on mobile and IoT devices, 2019, [online] Available at: https://www.tensorflow.org/lite [accessed 2 December 2019].

[32] C. Zhang, P. Patras, and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," arXiv preprint arXiv:1803.04311, 2018.

[33] R.S. Sandhu, "Role-based access control," Adv. Comput. 46, pp. 237–286, 1998.

[34] E. Yuan, J. Tong, "Attributed based access control (ABAC) for Web services," in: IEEE Int. Conf. Web Serv., IEEE, 2005.

[35] A. Ouaddah, H. Mousannif, A. A. Elkalam, A. Ait Ouahman,

[36] "Access control in the Internet of Things: Big challenges and new opportunities", Computer Networks 112, pp. 237–262, 2017.

[37] A. Kalam, et al., "Organization based access control," in: IEEE 4th Int. Work. Policies Distrib. Syst. Networks, IEEE Comput. Soc, pp. 120–131, 2003.

[38] S. El Bouanani, M. A. El Kiram, O. Achbarou and A. Outchakoucht, "Pervasive-Based Access Control Model for IoT Environments," in IEEE Access, vol. 7, pp. 54575-54585, 2019. doi: 10.1109/ACCESS.2019.2912975.

[39] M.R. Abdmeziem, D. Tandjaoui, I. Romdhani, "Architecting the Internet of Things: State of the Art", Robots and Sensor Clouds. Studies in Systems, Decision and Control, vol 36. Springer, 2015.

[40] C. Bormann, M. Ersue, A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, May 2014.

[41] A. Ouaddah, A. Abou Elkalam and A. Ait Ouahman, "FairAccess: a new Blockchain-based access control framework for the Internet of Things", Security and Communication Networks, pp. 1-22, 2017.

[42] ISO/IEC JTC 1, Information technology, iso/iec 29146:2016, A framework for access management, 2016.

[43] eXtensible Access Control Markup Language (XACML) Version 3.0, OASIS Standard, January 2013.

[44] A. Ameziane El Hassani et al., "Integrity-OrBAC: a new model to preserve Critical Infrastructures integrity", International Journal of Information Security, Springer, vol. 14, Issue 4, pp 367–385, 2014.

[45] A.E. Kalam, Y. Deswarte, "Multi-Orbac: a new access control model for distributed, heterogeneous and collaborative systems," in: 8th IEEE International Symposium on Systems and Information Security, p. 1, 2006.

[46] A. Abou El Kalam, Y. Deswarte, A. Baïna, M. Kaâniche, "PolyOrBAC: a security framework for critical infrastructures," Int. J. Crit. Infrastruct. Prot. pp. 154–169, 2009.

[47] I. Bouij-Pasquier, A. A. El Kalam, A. A. Ouahman, and M. De Montfort, "A Security Framework for Internet of Things," Springer International Publishing, pp. 19–31, 2015.

[48] M. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, P. Sheth, "Machine learning for Internet of Things data analysis: A survey". Digital Communications and Networks. 2017.

[49] U. Nations, "World urbanization prospects: The 2014 revision, highlights," Department of economic and social affairs. Population Division, United Nations, 2014.