**IoT’s Adaptable Resource Access Authorization: The Need for The Context**

**Mouiad Al Wahah\* Csilla Farkas\***

**\*Department of Computer Science and Computer Engineering, USC, Columbia, SC, 29205, USA**

**Abstract**

In the dynamic environment of IoT, the cooperating entities within a specific session cannot be determined ahead of time because of the entities’ mobility, environmental changes, and resources’ variability and their unpredictable availability. This situation requires that resource access control methods be suitably adaptable to the changing status of the cooperating entities and their dynamic environment. Traditional access control policies, like role-based access control, cannot be used in such situations because they presume beforehand the awareness of the users’ roles. Even the more advanced attribute-based access control approach does not solve this problem without making some modifications to its original formal settings since it depends on the attributes of the communicating entities to specify access control authorization. This paper describes an approach that highlights the need for the context in building adaptable access control policies for IoT’s resources in such a way that makes these resources, relatively, insurmountable against unauthorized access attempts. The paper develops a lightweight context ontology, CTX-Lite, that serves as a core ontology for context handling operations. The approach described in this paper uses the highly expressive capabilities of description logic (DL) and its reasoning power represented in OWL2 ontologies and Pellet reasoner and logic programming (LP) for modeling contexts and context-based access control rules. The LP is achieved using generic Jena rules and Jena reasoning engine.

**1 Introduction**

Everyday expansion of IoT perimeter through adding more devices, their increasing computing capabilities, their decreasing sizes, their connectivity, mobility, dynamicity, interoperability and collaboration among each other have added more services and abilities to our communities. However, this broad accessibility comes with great security challenges. These security challenges are represented by the difficulty of adopting ad-hoc access control policies for protecting IoT’s resources because of the variability and mobility of the IoT’s devices or the users of these devices. For example, old-styles access control policies like Mandatory Access Control (MAC) and Discretionary Access Control (DAC) have their shortcomings in this regard. MAC assumes awareness of security levels (clearance and classification) of the subjects and objects of the communicating entities in addition to the severe restriction imposed by MAC which makes it unsuitable for open environments like IoT. In DAC, the problem of multiple copies of the same information results in an uncontrollability over the protected objects and turns it unpractical for IoT dynamic world.

Generic Role-based access control (RBAC) cannot be used in this case since it assumes a priori the awareness of collaborating entities. Even the more advanced attribute-based access control does not solve this problem without making some modifications to its original formal settings.

In its nature, the contextual information is dynamic, and its values change over time, even during communication sessions. In such dynamic environments, we emphasize that an access control (AC) solution for IoT devices should be context-aware; it should react not only to observable value changes of individual contextual attributes, but also to the composite effect that is caused by multiple contextual attributes and their value changes. Also, users’ and devices’ mobility further exacerbate the AC challenges in IoT’s environment. Mobile users with known/unknown devices move in/out of the environment without the protection of infrastructure-based firewalls exposes the environment to more security threats and attacks. So, an effective context-aware access control solution should take into account the level of confidence in the entity trying to gain access to sensitive resources as well as the confidence level in the provided contextual information.

The main contribution of this paper is to develop OWL ontologies for representing the contextual information and context-based access control (CBAC). More specifically, this paper describes an approach that highlights the need for context in building adaptable access control policies for accessing IoT’s resources in such a way that makes these resources, relatively, secure against unauthorized access attempts. The paper develops a lightweight context ontology, CTX-Lite, which wraps the low-level contextual information, raw data provided by sensors, RDF stores, SPARQL endpoints, etc., and provides the annotation services necessary for getting high-level context from those raw data observations. Hence, CTX-Lite ontology serves as a core ontology for context handling operations. The approach described in this paper uses semantic web technologies, specifically OWL2, Pellet reasoner, Jena inference, and generic Jena rules.

**2** **Related Works**

Despite the research on the adaptability of access control models for dynamic environments like IoT is still in its preliminary stage, yet some efforts have been made to address this problem. Most of the access control approaches proposed so far are based on ontologies for modeling the context and access control for security adaptation. In this section, we are not going to give a complete list of these efforts. Rather, we focus on those approaches that either use the context as the main principle for access authorization in dynamic environments, or those which address the need for the context in making access authorization decisions.

Of the first mature research work in context-based access control is that of A. Corradi et al. [6] which adopts a context as a principle for security policy specification and enforcement process. Unlike the traditional access control model which is user-centric, access permissions in CBAC are directly associated with contexts. They adopt an RDF-based format for context representation to cover heterogeneity of data representation. However, it does not extend the RDF-based format as a means to infer the relationships of entities and this was a limitation since the approach does not support the deduction of higher-level context from the primary sensed context.

A. Toninelli et al [7] introduce the Proteus framework. The framework activates the context based on description logic (OWL ontologies) and logical programming rules to enable the dynamic adaptation of policies by linking the requester to the data and the context. The proposed framework aimed to work in pervasive and ubiquitous environments. Based on Proteus framework, P. Bellavista and R. Montanari [8] proposed an implementation for IoT adaptive context-based access authorizations. However, using only OWL ontologies encounters some difficulties with regard to the definition of some kinds of policies—specifically those requiring the definition of variables. For example, the use case of a “same location policy”, that grants access to files only when is a good example that illustrates the need for variables [19].

A. Dersingh et al. [9] have proposed a policy system that separates context’s management from access control management, it is relatively similar to our approach in this concern. The policy system they proposed extends the eXtensible Access Control Markup Language (XACML) by adding the capability of using context vocabularies in the policy and designating subjects and resources via semantic knowledge.

L. Seitz et al. [10] described a framework for authorization and access control on IoT, in the context of interconnected systems consisting of resource-constrained devices not directly operated by humans. Their approach, however, does not support context nor semantic-web technologies and works as an add-on for XACML which makes it difficult to adopt for IoT because evaluating XACML policies is too heavyweight for constrained devices and this is why the authors in [10] made the authorization decision process external to XACML policies.

H. Mohamed et al. [11] present a semantic-based authorization approach for controlling access in collaborative cloud environments. Their approach is also based on the XACML architecture and makes access decision according to contextual situations. Authorization context is evaluated by XACML Engine. The contextual information is retrieved from several sources, i.e. by using attribute finders to dynamically search, and perform dynamic queries for the environmental values of these attributes. However, all the three approaches, [9, 10, 11] highlight extending the well-known ABAC XACML framework with semantic-based context.

N. K. Sharma and A. Joshi [12] have shown that OWL ontologies and their DL-based reasoning are able to model and enforce ABAC model, which in turn, can be used to enforce DAC, MAC, Flat RBAC and Hierarchical RBAC based policies. Bilal S. et al. [13] proposed some context-based access control policies for Android OS. These policies restrict the applications from accessing a specific resource based on the user context, but their approach, though is so well scalable, does not support semantics and, hence, is not suitable for IoT’s dynamic environment.

M. Covington and M. Sastry [14] presented a contextual attribute access control (CABAC) model which was realized in mobile applications. They used contexts as add-ons for the ABAC.

A. Kayes et al. [15] developed a context-aware access control following the role-based access control mechanism. Their method considers context-aware user-role and role permission assignments and consequently supports context-specific access control to resources by leveraging the dynamically changing context information. For their approach, they use OWL and SWRL for modeling and reasoning about access control policies. Despite they solved dynamic context information, but still their approach does not solve the inherent problem of any RBAC-based policies which is the exponential expansion of roles.

Tim Finin et al. [16] proposed a method for context-sensitive access control for IoT that uses high-performance EYE reasoner for reasoning over context, CBAC, and rules ontologies. These OWL ontologies were formulated using N3 style to be more user-friendly and compatible with EYE reasoner [17]. Their approach focuses on extending ABAC policies by encompassing contexts with these ABAC policies. One of the main advantages of ABAC is requesters do not have to be known a priori by targets, providing a higher level of flexibility for open environments, compared to RBAC models. Nevertheless, in ABAC everyone must agree on a set of attributes and their meaning when using ABAC, which is not easy to accomplish [18], especially in IoT environment. R. Dautov et. al. [20] highlight the importance of reasoning in OWL and SWRL reasoning in leveraging this reasoning ability in enforcing access control policies. They didn’t propose any specific method for exploiting these abilities but they describe general ideas of how to use reasoning in OWL and SWRL for access policy enforcement.

This paper’s approach is similar to the work done by Tim Finin et al. [16]. They also formulated a method for context-sensitive access control for IoT’s resources. However, the approach taken in this paper takes a different direction. First, this paper deals with the context as the primary component to specify access control policies. We are separating the contextual information related to subjects, objects, environment…etc. from access policy, while they use context as an add-on for attribute-based access control. Second, in terms of performance, they use high-performance EYE reasoner for reasoning over context, CBAC, and rules ontologies to get high-level reasoning performance. In our approach, we aim at using light-weighted OWL ontologies for representing contexts and access control policies and to benefit from full-blown OWL2 DL-based reasoner Pellet for getting inferences. We use the full-blown tableau reasoner Pellet (despite its being slow in comparison with rule-based EYE reasoner) to reduce the dependency on rule-based Jena inference reasoning and use them exclusively with specific conditions like when we need to use variables, for example.

**3 Formal Representation of the Contexts**

It is agreed upon in the research community [2] that the first research on context-aware computing was done in 1992 at Olivetti research Ltd, England [3]. Want et al. have developed Active Badge Location System (ABLS). The badge is used for tracking the locations of the company staff members. After that, Schilit et al. [24] categorized contextual information into three categories depending on three common questions that can be used to determine the context. These questions are commonly referred to as a 3Ws context model. These questions are:

1. Where the agent (user) is: This includes all location related information such as GPS coordinates, common names (e.g., hospital, university, shopping center), place names (e.g., College of engineering, Fire Department), specific addresses like 1230 Pendleton street, etc.
2. Who is the agent with: The information about the agents present around the concerned agent.
3. What resources are nearby: This includes information about resources available in the area where the agent is located, such as machinery, smart objects, and utilities.

Dey defines the context as any information that can be used to characterize the situation of an entity (person, place or object) [2].

Many other researchers in different scientific and academic fields have already developed other context definitions and models. For this paper, we accept Schilit et al [24]. categorization of contexts and we add to it another question for temporal context, for example: when the agent encounters with this context? We believe that these four categories for the context are the most fundamental information to capture the context of communicating agents. For this paper, we define the context as follows:

**Definition 1 (Context).** The context is a set of observations that relevant to the state of the entities/agents incorporated in specific communicating sessions under which the access policy decision is made. A typical decision is a “permit” or “prohibit”

**Definition 2 (Observation).** An Observation is a situation in which a sensing method has been used to estimate or calculate a value of a property of a feature of interest.

Links to sensing and sensor describe what made the observation and how; links to property and feature detail what was sensed; the result is the output of a sensor; other metadata details times, locations, …etc. [4].

We divide contextual information into two types, raw observation , and composite observation . The context, in general, will be a collection of some raw observation and some composite observation . By raw observation we mean any atomic, indivisible attribute of an entity like a specific point in time, longitude …etc., which usually fed by only one sensor, provided by a user, one entry from a DB table, or a node of an RDF graph. The composite observation is that observation which is constructed from raw observations in terms of logical operation AND. So, the observation O is represented as

, or

, as in

We use the term observation, not the “situation”, to emphasize the visibility of the attributes of the collaborating entities and hence the more visible attributes that can be seen (sensible) by the access control policy, the broader the context will be and more intact and precise access decision to reach.

For any context , there will be a set of raw observations and/or composite observations that constitutes . The concept of the context is built by applying a set of logical operations or by constraining contextual properties. These operations and constraints are collectively called the constructors in paradigm, and are given (along with their equivalent and statements) by:

|

|

|

|

|

Where is a context, is a composite observation, is a nonnegative integer, are class instances (constants in ) and is a property name (predicate in ), is for universal set and is for the empty set.

**3.1 The Lightweight Context Ontology CTX-Lite**

World Wide Web Consortium, W3C[[1]](#footnote-1) published a list of ontologies which have been distinguished as “good ontologies”. These ontologies have common prerequisites that make them “good”, according to W3C. These prerequisites are: fully documented, dereferenceable, used by independent data providers, and possibly supported by existing tools [21]. Other researchers also put their prerequisites for ontologies to be good ontologies via balancing between the complexity of the ontology and the purpose it is built for. For example, W. Wang et. al. [22] depend the principles of lightweightedness, completeness, compatibility, and modularity as the fundamental prerequisites for the ontology to be acceptable in open-linked data environments. In their modeling for IoT-Lite, M. Bermudez-Edo et. al. [23] take more extreme view by defining 10 rules that the ontology should adhere to be good and scalable.

For our lightweight context ontology CTX-Lite, we depend on the following prerequisites for reasons clarified below:

* Validity, the most important feature of ontology is to be valid. Without being valid, any ontology will be useless.
* Interoperability, the CTX-Lite ontology should be interoperable with other standard ontologies. There is no ontology that can fit all the computing domains and there must be an ability to import, reuse, and interoperate with other ontologies from different domains.
* Genericness, CTX-Lite ontology does not intend to be a comprehensive or complete. Rather, it is aimed to be a core ontology for modeling contexts. If the domain of application requires specialization and/or generalization to the vocabulary of the CTX-Lite ontology, the ontology designer can add to this ontology any concept to specialize the contextual information, or he/she can use the concepts defined in CTX-Lite as a generalization for high-level contexts representation.
* Simplicity, simple ontologies easily find their ways to be adopted by other applications. For example, Friend Of A Friend (FOAF) ontology[[2]](#footnote-2) is being used by many projects and is existing in so many websites like Live Journals and academic sites because of its simplicity (only 19 classes, 44 object properties, 27 datatype properties).
* Scalability, the ontology should be scalable to a large volume of data instances.

The context ontology, CTX-Lite, defines eight basic concepts as top-level concepts, these concepts are Agent, Device, Service, Network, Location, Time, Activity and Context. The ontology has the concept as the top concept. The Context concept represents the higher contextual concept and it has two subcontexts for representing raw context and composite context, and these are RawCtx and CompositeCtx, respectively. The CompositeCtx has seven subconcepts that represent our four fundamental high-level contexts as well as other three concepts for handling unnamed contexts, personal contexts and spatiotemporal contexts. These sub concepts are: Behavioral, Temporal, Spatial, Spatiotemporal, Environmental, Personal, and Unnamed concept, figure (1) shows main concepts’ hierarchy. The CTX-Lite ontology is only 140 triples and this is because we aim at making it as generic as possible and to be used as a core ontology for context-based applications. Also, being lightweight, CTX-Lite ontology makes it possible to make fast annotation and efficient processing time.

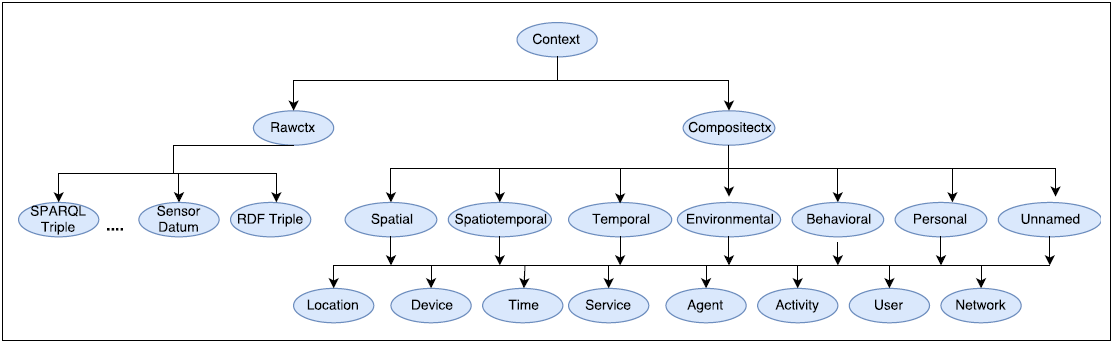


Figure (1) CTX-Lite Ontology Main Concepts Hierarchy.

For validity, we verified the validity of CTX-Lite ontology using Protégé[[3]](#footnote-3) and Jena Framework[[4]](#footnote-4). In both cases CTX-Lite proves its validity.

For interoperability, we linked CTX-lite ontology to other well-known standard ontologies (e.g. Time, SSN, geo, …etc.), scenario 1 below will illustrate this characteristic.

For genericness, the CTX-Lite ontology does not specialize any context and leave this for the domain that embraces it for its knowledge representation. For simplicity, CTX-Lite ontology includes only 29 classes, 7 data properties, and 15 object properties, and only 140 triples.

For scalability, we take into account that the domains which make use of ontologies are different but they still have in common general concepts that they share. At the same time, every domain has some very specific ontologies that are exclusive to that domain. Hence, CTX-Lite ontology aims at providing the general concepts in context-based access control applications and leave those domain-specific concepts and properties. This design philosophy will significantly help to control the size of contextual datasets.



Figure (2) Context Ontology Components.

**Scenario 1**

Most modern cars are equipped with a variety of sensing devices. For example, most cars have sensor units such as Air Bag Sensors, Blindspot Sensors, Back up Sensors, Low Air Sensors, …etc. Also, some cars are already equipped with Bluetooth and WiFi devices. For example, starting in mid-2014 and working through its OnStar division, GM introduced built-in 4G LTE WiFi provided by AT&T and launched the service in certain 2015 model-year vehicles which permit up to 7 devices to be connected within a car [1]. In this ongoing progress, we expect that by 2021 all cars will be equipped with WiFi and Bluetooth devices and become connected with the external world. From security and privacy point of view, we are concern about protecting a vehicle and its user information from a malicious use or unauthorized access. Now consider the following scenarios:

Alice has a car equipped with a WiFi connection and a built-in GPS device. By default, all operational units in the car, according to the manufacturer setting, are accessible by the car driver (or owner of the car) and hence in normal daily commuting, the network connection is provided for Alice by default, but it is forbidden for other car riders. When there is an idle system functionality, these idle systems should be accessible to the car passengers. For example, when on a highway and the car speed is above 70 kmph and the GPS device readings state that the destination is to be reached in no less than 30 minutes. Also, there is, other than Alice, another rider in the car, then access control should permit the other car riders to access to the WiFi connection so that they can use their devices by connecting them to the car WiFi system.

The above scenario explains two desired properties of access control policies, the first one is the normal behavior in normal driving conditions when access to the car WiFi connection is permitted only for the car driver. The other is the adaptive behavior which depends on several conditions.

For example, to represent the context of the passenger presence (sitting) on the car seat

, the axiom equivalent to it is shown below:

Note that the raw observations in this example are:

, while the axiom as a whole represents the context .

This says that the context of that is not the driver seat, is represented by an entity that has a device attached to it. This device is a sensing device which makes an observation. The observation is related to a property to be observed, which in its turn, is associated with a feature of interest that’s to be observed or estimated. The feature of interest is related to an observation result. This result must have a value, which in its turn has a unit to measure.

In logic programming ( which is part of paradigm, where all the variables are instantiated over the knowledge base ()), this context can be written as:

Now let’s analyze our example and see how to annotate, instantiate, and infer this context. The ssn ontology[[5]](#footnote-5) presents the class ssn:SensingDevice as a subclass of ssn:Sensor as well as of ssn:Device. The OccupancySensor is a very specific type of sensing devices. We first create a small hierarchy of sensing device types. It specifies that OccupancySensor is a sensing device and that CNDtekD100 is an instance of this sensing device. Now suppose we have a car Smart-BuickCar-2016. The car is provided with BuickPassengerSeat as passenger seats. This BuickPassengerSeat is equipped with a sensing device, CNDtekD100, for sensing and estimating the human-bodyweight (the property to be observed), and the FeatureOfInterest (weight). This requires that the obersv1 (observation), observ-value (observation value), so (sensor output), dataValue and the unit of measure to be annotated also. Figure (3) below illustrates an excerpt of the annotation of the context ontology with other well-known ontologies. The ssn, qu[[6]](#footnote-6), iot-lite[[7]](#footnote-7), units[[8]](#footnote-8) ontologies are imported to help in carving for online semantics that support the context ontology, CTX-Lite, we build.

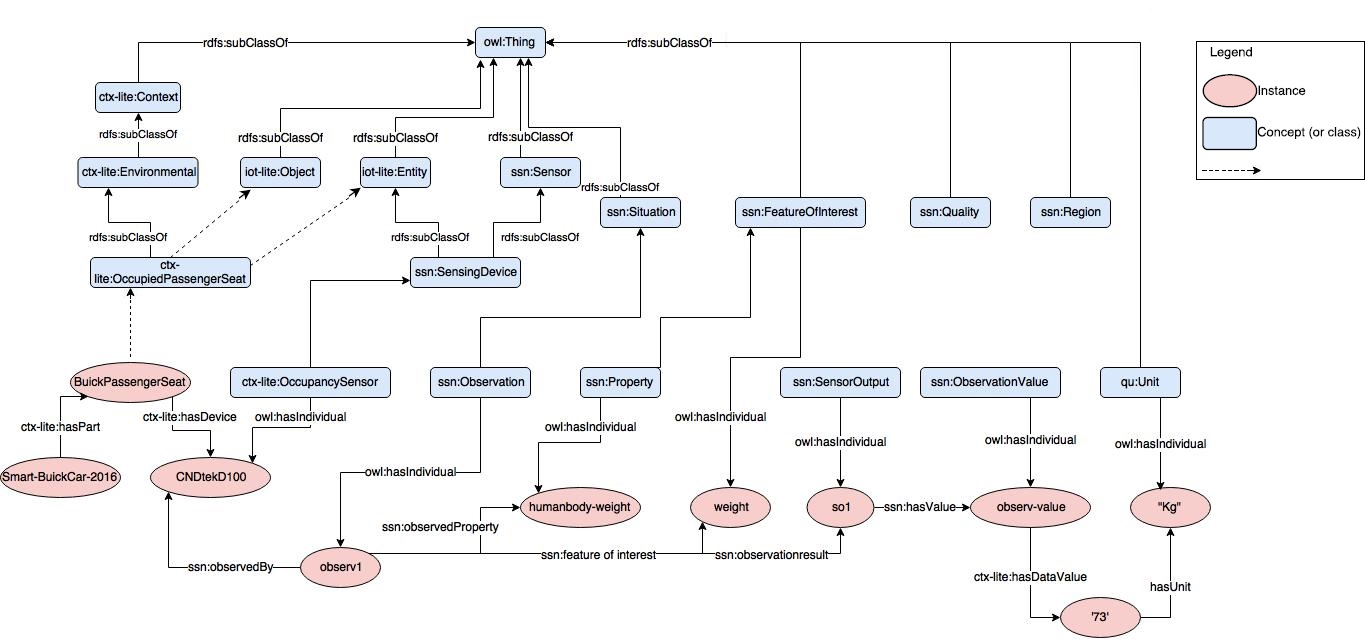


Figure (3) excerpt of the annotation process using CTX-Lite ontology with other well-known ontologies

**3.2 Spatiotemporal Contexts Representation**

Spatial and temporal contexts represent the most permanent components of any contextual information as they provide the gluing joints of any context. Events in the world occur at specific locations and times and usually have finite durations. In our model, we capture the user’s (agent) location as geospatial coordinates represented as longitude and latitude (any other method also works like GPS coordinates). The important feature of the geospatial coordinates is that they can flexibly be mapped into other location representations. For example, geospatial coordinates of latitude 33.991981 and longitude -81.031345 points to the address 555 Assembly street, Columbia, SC, USA (address), which is also the address of Storey Innovation Center (building).

In our model, time is captured using the Time ontology[[9]](#footnote-9). Time concepts, like space, can be mapped into conceptual concepts, like during, before, after…etc., or to concrete concepts like 12:56:33 EST. The core classes and properties of the Time ontology are shown in figure (4), as they have been published by W3C.

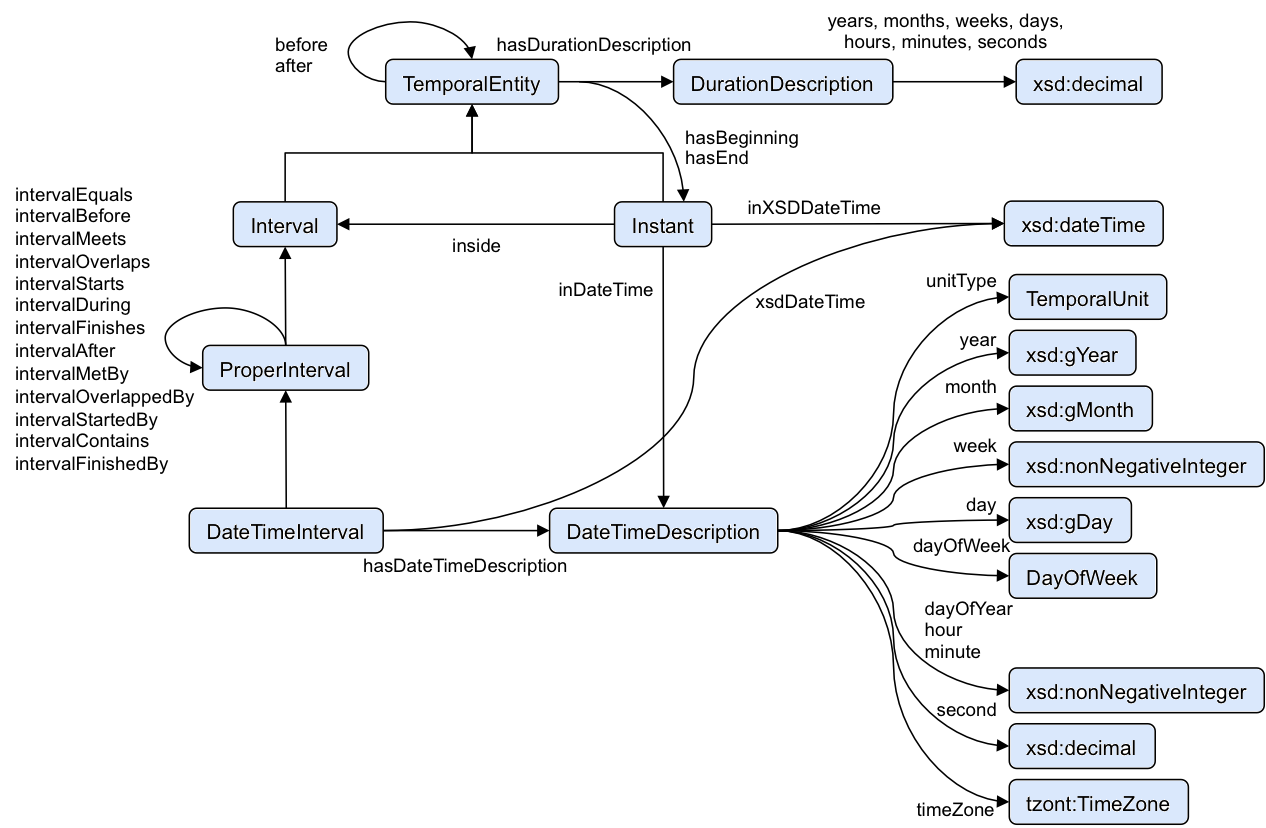


Figure (4) Time ontology.

Using Allen’s 13 pairwise disjoint temporal relations [5] and Time ontology, an interval-based and point-based representation of time is possible and straightforward. Point-based representation assumes linear ordering of time points with three possible relations, namely , , which referred to as , , and , respectively. Based on these ordering relations, intervals can also be defined as ordered pairs of points and with , which often referred to as and of an interval.

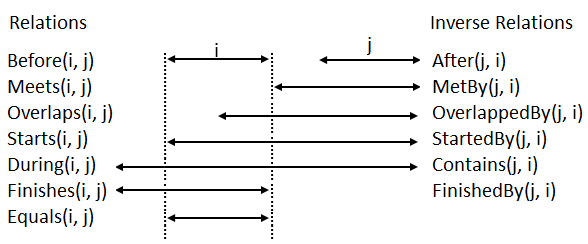


Figure (5) Allen’s Temporal Relations [5].

For example, for representing time instants, we use the class and the of Time ontology. Figure (5) below shows how to represent the instant for 15:32:07, August 23rd, 2017.

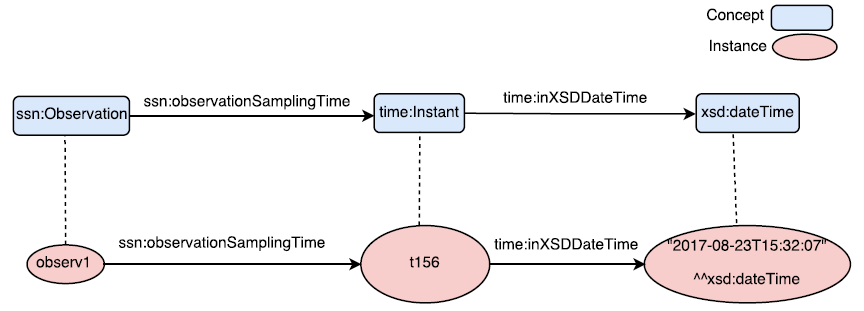


Figure (5) instant for 15:32:07 on August 23rd, 2017 using Time ontology.

**4 Modeling Context-Based Access Control Policy**

We do not depend on any predefined access control paradigm. Rather, we adopt the context as the core element around which all access authorization operations are based. In our approach, the context-based depends on the visibility of certain contexts that to be fed to the access control policy. In its turn, the access control policy, depending on the provided context information, may permit or prohibit such access authorization request. We use semantic web technologies to allow representing and reasoning about the contexts as well as the access control policies themselves.

In the proposed approach, the context-based access control model, or CBAC, consists of a set of rules and each rule has the following form:

Where,

= Request,

= Request type,

= Request subject,

= Subject[[10]](#footnote-10) context, or context of the agent working on behalf of a subject,

= Request resource,

= Resource (Object) context, or context of the agent working on behalf of the resource, or the resource’s owner context,

= Request decision,

= Decision effect, and

=General context

Each rule is triggered by an access request and is evaluated to a decision at runtime. The decision has an effect which, in our model, is either a prohibit or a permit. The star on the is to indicate that this field is optional while the other fields are mandatory.

Then we can represent the as a set of rules as the one given above:

=

Now let’s represent our access control model using . Given the above clarification of how to represent rules, the rule axiom becomes straightforward and is given as:

In OWL 2, this is written as:

When all the access control rules are specified as we have done above, run-time evaluation of the access authorization is accomplished when a request received by access control policy engine.

**Scenario 2**

Martha is a 70-year-old woman and she lives alone in her house that lies outskirts of Columbia City, SC. Martha has some health problems like sudden hypertension, fluctuations in heartbeat. Concerned about the possibility of falling, she begins to use a smart accelerometer device with heartbeat and blood pressure monitors embedded in it. The device transmits its data, the x-y-z coordinates, the systolic and diastolic blood pressure, and the heartbeat of the user as continuous streams to a nearby laptop using WiFi signals and then from that computer the streams are transferred to the health agency main server or into the cloud store. Martha has given the laptop computer the permission to dial the emergency number, 911, on her behalf, in case some urgent medical condition arises.

To access Martha’s medical data stored on the laptop, any paramedic must be on duty at the time of any emergency that may happen to Martha. For example, a severe drop in the heartbeat, a fall accident…etc. One day, Martha has fallen and went into an emergency condition. According to this context, Martha’s laptop should be able to recognize this context and dial the 911. This is case one that illustrates the need for context in such situations. Case two, suppose now that Martha’s laptop has dial 911 and a paramedic Joe has arrived to help. The InEmergency context is already recognized but the system needs to link Joe’s OnDuty context with Martha’s InEmergency context so that Joe can access Martha’s medical data resides on the health agency server or on the Cloud.

From the above scenario we build two access control authorization rules shown in table (1).

Table (1) access authorization rules for scenario 2 example.

|  |  |
| --- | --- |
| No | Rule |
| Rule#1 | An owner of a device can access the device’s resources irrespective of her/his context. |
| Rule#1 | A paramedic personnel who is on duty at the time of emergency can access patient devices’ resources in case that patient suffers from that emergency case. |

The first rule does not encompass any context and hence there is no need for complex rule. The following simple Jena rule can serve this purpose:

The only note to be mentioned here is that we need a unique identifier[[11]](#footnote-11) that peculiarly identifies each owner which can tell who the real owner of the resource, in our rule above that identifier is “PAT29013”.

For the second rule of the policy we need two contexts, one associated with the subject and the second is associated with resource’s owner (not the object). Let’s build the context for Martha’s health condition using our approach:

And the context is given by:

Associating the context for Martha with context for Joe will lead to a permit impact for Joe’s request to access Martha’s health records using the following rule:

Again, we use unique identifiers, “MSS56006” and “PAT29013” for associating each context with the assigned entity. Now the access control policy engine can query the KB to know which has access right to what. For example, using the following SPARQL query:

And the result is:

**------------------------------**

**| Request | Effect |**

**==================**

**| app1:r1 | cbac:Permit |**

**-------------------------------**

So, and as shown above, access to a resource is granted only when the subject’s context, the resource’s context (or the resource’s owner as it is the case in scenario 2), and/or any other context that is specified by the access control policy match a particular “Permit” rule.

Figure (7) illustrates the flow of reasoning process between context manager (DL reasoning engine) and the access authorization module (Jena inference engine).

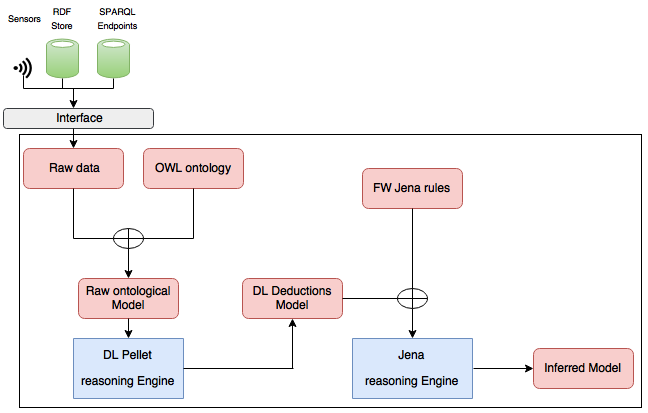


Figure (7) illustrates the flow of reasoning.

And figure (8) represents the architectural view of the proposed method.

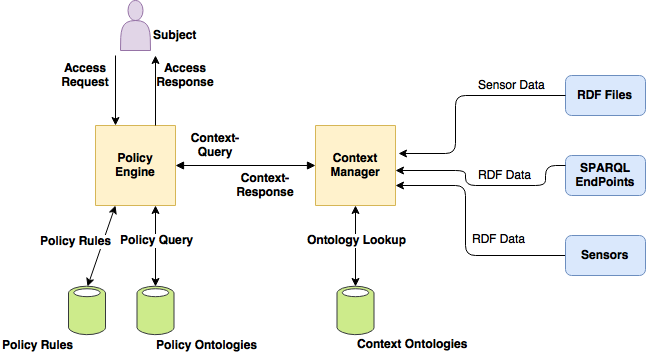


Figure (8) architectural view of the proposed method.

We have decoupled the context management from access control policy engine. This decoupling has a number of advantages:

* It permits to reuse the context and its management in other applications and not limiting their use for only access policies applications.
* It allows distributing the processing overhead of both access control management and context management as it is required by the application domain.
* It permits the access control engine to adopt multiple access control paradigms at the same time. For example, when used in domains that need adaptable behavior like IoT, or can be tailored to work on domains that regard RBAC as the method of choice.

**5 Implementation and Data Sets Used in the Approach**

All experiments were conducted using two laptops, PC1 is a Windows 8.1, 1.9 GHz CPU, 4 GB RAM and 500 GB HDD, while PC2 is Windows 10 3.8 GHz CPU, 32 GB RAM, and 2 TB HDD. Ten sets of data have been generated using online data generator[[12]](#footnote-12)and been processed to remove duplicates. Then we used Cellfie[[13]](#footnote-13) plug-in for Protégé to assert these instances to the model ontologies. The sets are 10000, 20000, 30000, 40000, 50000, 60000, 70000, 82000, 90000, and 100000 triples in their size. The reasoning process is accomplished using DL reasoner Pellet while LP is accomplished using forward chaining Jena inference engine and user-defined rules. The DL reasoning has been used for checking ontology satisfiability and consistency, conceptual subsumption, and instance realization. The bulk of reasoning process in this paper is done by DL, while LP is used only for simple situations like in case of need to variables.

The reasoning process has been repeated ten times for every one of the mentioned data sets on each one of the two PCs. Then we take the average time computed for these ten repetitions for reaching concrete values for the reasoning process execution time. The same thing is repeated in calculating SPARQL query response time but for bigger data sets because these are the results of inference upon the original data set models. Figures (9-*a*), (9-*b*), and (9-*c*) represent the results of DL reasoning time, DL+PL reasoning time, and SPARQL query response time, respectively.



Figure(9-*a*) DL reasoning time.



Figure(9-*b*) DL+PL reasoning time.



Figure(9-*c*) response time of SPARQL query upon inference model.

There are some points to hint on regarding reasoning and the deployment of the proposed method. These are:

* In addition to using lightweight ontologies for modeling context and access control operations, complex DL axiom are computationally expensive and are not feasible for IoT resource-constrained devices. It is important to balance the tradeoffs between the expressivity and complexity on one hand, and the scarce computing power of IoT devices on another hand.
* Reasoning in, general, and DL reasoning, in special, is very resource-consuming (both in time and space). Hence, it is not practical not only for small devices but even for medium-level computing devices like smartphones and average PCs to implement the reasoning process themselves. Instead, it is more feasible to dedicate a much powerful machine to accomplish the reasoning process and the result of inference have to be reachable for access control engines, as a service for example.
* It is also more productive for these machines to employ persistent reasoning, in which the reasoner saves the inferred information together with its internal state into a file, which can then be reloaded with much less computational effort than reasoning would require if it begins from scratch, or they can use incremental reasoning in which the reasoner carefully determines which parts of the precomputed inferences may be affected by an incoming change and only recomputes a subset of the inferences [25, 26].

**6 Conclusion and Future Works**

In this paper we have shown why dynamic environment of IoT necessitates the adoption of context-based access control. We have modeled such context-based access control using lightweight OWL ontologies with help of Jena generic rules. The ontologies we have written represent the formal specification of the contexts and the CBAC. The reasoning process we adopted is based on DL and LP. All access authorizations operations suggested in this paper are determined based on contexts of the main communicating entities.

As a future work, the reasoning process and the complexity of the axioms employed in context ontology as well as CBAC ontology are subject to further analysis. We are planning to engage obligations (as we did not do so in the current study), which are conditions that must be met before access authorization is granted an access to a resource, and commitments, which are conditions that must be respected after access to a resource is granted to the requester. We believe that obligations and commitments are two challenging issues in the world of IoT’s access control authorization. Another challenging problem we are considering for future work is the possibility of integrating human activity pattern recognition using machine learning with the ontological-based CBAC for smart home security. Currently, our proposed method provides CBAC but it does not have the ability to recognize human activities, so integrating robust machine learning techniques, like Bayesian network models, by feeding their outputs to our model, will greatly enhance context recognizability in smart home security frameworks.

**References**

1. 4G LTE In-Car Wi-Fi: Features & Information | Buick. 2018. Retrieved January 26, 2018 from <http://www.buick.com/4g-lte-in-car-wifi.html>
2. G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, “Towards a better understanding of context and context-awareness”, in Handheld and ubiquitous computing, pages 304–307. Springer, (1999).
3. R. Want, A. Hopper, V. Falcão, and J. Gibbons, “The active badge location system”. ACM Trans. Inf. Syst., 10(1):91–102. (1992).
4. W3C Semantic Sensor Network Incubator Group. 2008. Retrieved December 11, 2017 from <https://www.w3.org/2005/Incubator/ssn/>
5. James. F. Allen, “Maintaining Knowledge about Temporal Intervals “Communications of ACM, Vol 26, no 11, November 1983
6. Corradi, R. Montanari, and D. Tibaldi, “Context-based Access Control for Ubiquitous Service Provisioning”, pp. 444-451, 28th Annual International Computer Software and Applications Conference (COMPSAC'04), 2004.
7. A. Toninelli, R. Montanari, L. Kagal, and O. Lassila, “Proteus: A semantic context-aware adaptive policy model,” in Eighth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY’07), pp. 129–140, IEEE, 2007.
8. Bellavista, P. and Montanari, R., “Context Awareness for Adaptive Access Control Management in IoT Environments”, in Security and Privacy in Cyber-Physical Systems: Foundations, Principles and Applications (eds H. Song, G. A. Fink and S. Jeschke), John Wiley & Sons, Ltd, Chichester, UK. doi: 10.1002/9781119226079.ch8, 2017.
9. A. Dersingh, R. Liscano, and A. Jost, “Context-aware access control using semantic policies,” in Ubiquitous Computing and Communication Journal (UBICC) Special Issue on Autonomic Computing Systems and Applications, vol. 3, pp. 19–32, 2008.
10. L. Seitz, G. Selander, and C. Gehrmann, “Authorization framework for the Internet-of-Things,” in 2013 IEEE 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2013), 2013.
11. H. Mohamed, C. Abdelghani, W. Thierry, D. Karim, “Semantic-Based Authorization Framework for Multi-Domain Collaborative Cloud Environments”. Procedia Computer Science. 109. pp 718-724, 2017.
12. N. K. Sharma and A. Joshi, “Representing attribute-based access control policies in owl,” in 2016 IEEE Tenth International Conference on Semantic Computing (ICSC), California, USA, pp. 333–336, 2016.
13. B. Shebaro, O. Oluwatimi, E. Bertino “Context-based Access Control Systems for Mobile Devices” DOI 10.1109/TDSC.2014.2320731, IEEE Transactions on Dependable and Secure Computing, 2014.
14. M. Covington and M. Sastry. A contextual attribute-based access control model. In Proceedings of the 2006 International Conference on the Move to Meaningful Internet Systems (OTM), pages 1996–2006. Springer, 2006.
15. Kayes, A. S. M. & Han, Jun & Colman, Alan, “A Semantic Policy Framework for Context-Aware Access Control Applications”, 10.1109/TrustCom.2013.91, 2013.
16. P. K. Das, S. Narayanan, N. K. Sharma, A. Joshi, K. Joshi, and T. Finin, “Context-sensitive policy based security in internet of things,” in Smart Computing (SMARTCOMP), 2016 IEEE International Conference on. IEEE, pp. 1–6, 2016.
17. R. Verborgh, and J. De Roo, “Drawing conclusions from linked data on the web”, IEEE Software 32, 5 2015 (May).
18. L. Ramos, J. Bernabé, A. Gómez, I. Maestro, M. Nati, N. Gligoric, SocIoTal Deliverable. D2.2., “Framework Specification for Privacy and Access Control”, 2014.
19. Julian Schütte and Tobias Wahl. Description logics-based handling of inter-domain policy conflicts using meta-policies. IEEE Vehicular Technology Magazine, 5(3), 2010.
20. R. Dautov, S. Veloudis, I. Paraskakis, S. Distefano, “Policy Management and Enforcement Using OWL and SWRL for the Internet of Things”. In: Puliafito A., Bruneo D., Distefano S., Longo F. (eds) Ad-hoc, Mobile, and Wireless Networks. ADHOC-NOW 2017. Lecture Notes in Computer Science, vol 10517. Springer, Cham, 2017.
21. Good Ontologies - W3C Wiki. 2015. Retrieved December 27, 2017 from <https://www.w3.org/wiki/Good_Ontologies>
22. W. Wang, S. De, R. Toenjes, E. Reetz, and K. Moessner. “A comprehensive ontology for knowledge representation in the internet of things”, In Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on. IEEE, pp 1793–1798, 2012.
23. M. Bermudez-Edo, T. Elsaleh, K. Barnaghi, and K. Taylor, “IoT-Lite: A Lightweight Semantic Model for the Internet of Things. In UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld, 2016.
24. B. Schilit, N. Adams, and R. Want, “Context-aware computing applications,” in Mobile Computing Systems and Applications, 1994. Proceedings., Workshop on, December 1994, pp. 85 –90.
25. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. of Web Semantics 5(2), pp 51–53, 2007.
26. Yevgeny Kazakov, Pavel Klinov, The benefits of incremental reasoning in OWL EL, Proceedings of the 2013th International Conference on Posters & Demonstrations Track, p.57-60, October 23, 2013, Sydney, Australia

1. http://www.w3.org/ [↑](#footnote-ref-1)
2. http://www.foaf-project.org/ [↑](#footnote-ref-2)
3. https://protege.stanford.edu/ [↑](#footnote-ref-3)
4. http://jena.apache.org/ [↑](#footnote-ref-4)
5. http://purl.oclc.org/NET/ssnx/ssn# [↑](#footnote-ref-5)
6. http://purl.org/NET/ssnx/qu/qu# [↑](#footnote-ref-6)
7. <http://purl.oclc.org/NET/UNIS/fiware/iot-lite> [↑](#footnote-ref-7)
8. http://sweet.jpl.nasa.gov/ontology/units.owl# [↑](#footnote-ref-8)
9. https://raw.githubusercontent.com/w3c/sdw/gh-pages/time/rdf/time.ttl [↑](#footnote-ref-9)
10. In academia, we found that some researchers use the term subject to refer to the object’s owner, while the term requester is used to denote the entity that requests access permission to the object. In this paper, we use the term owner as it is, i.e., the object’s owner and terms subject and requester to mean the same thing which is the entity that requests access permission to the object or resource. [↑](#footnote-ref-10)
11. This paper assumes that there is an authentication method is in place and it does not authenticate users or agents requesting access, but it can use authentication identifier as a unique identifier for this purpose. [↑](#footnote-ref-11)
12. http://freedatagenerator.com/ [↑](#footnote-ref-12)
13. https://github.com/protegeproject/cellfie-plugin/wiki [↑](#footnote-ref-13)