SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2014; **7**:2696–2712
Published online 15 February 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.424

SPECIAL ISSUE PAPER

Context-aware usage control for web of things

Guangdong Bai[1], Lin Yan[1], Liang Gu[1,2†], Yao Guo[1]* and Xiangqun Chen[1]

[1] Key Laboratory of High Confidence Software Technologies (Ministry of Education), School of EECS, Peking University, Beijing, China
[2] Department of Computer Science, Yale University, CT, U.S.A.

ABSTRACT

The Web of Things (WoT), inherited from the Internet of Things (IoT), encapsulates functionalities into publishable services on the Web to enable the IoT a seamless integration with the Web. The openness of the Web, in turn, directly exposes WoT to existing attacks from the Web. In addition, WoT possesses characteristics of high security and privacy concerns, mobility, and limited capabilities, which require specific and additional security and privacy protection beyond existing mechanisms. More importantly, WoT is inherently connected to its context, so context information must be taken into account in its security and privacy measures.

To address these challenges, we propose a context-aware usage control model (ConUCON), which leverages the context information to enhance data, resource, and service protection for WoT. On the basis of ConUCON, we also design and implement a context-aware usage control framework on the middleware layer in our ongoing SmartHome project, to provide security and privacy protection. ConUCON is designed specifically to express the context-aware usage policy specification, such that security and privacy requirements can be easily specified and enforced with the proposed model and framework. Finally, we apply ConUCON to a remote appliance management prototype, as a case study, to demonstrates its feasibility in a real environment. Copyright © 2012 John Wiley & Sons, Ltd.

KEYWORDS

security; context awareness; usage control; Internet of Things; Web of Things

***Correspondence**

Yao Guo, Institute of Software, School of EECS, Peking University. Beijing 100871, China.
E-mail: yaoguo@sei.pku.edu.cn

†This paper was written when Liang Gu was a PhD student in Peking University. The revision was carried out when he became a postdoctoral associate in Yale University in 2011.

1. INTRODUCTION

Connecting isolated smart objects like sensors and actuators, ad-hoc networks as well as systems through unique addressing schemes, Internet of Things (IoT) [1] extends the traditional Internet from computer networks to an infrastructure integrating the physical world with computer networks. However, systems and networks in IoT are each designed and implemented into unique architectures and provide distinctive interfaces; significant effort and coordination is still required to connect them seamlessly. Web of Things (WoT) [2] solves this problem by introducing Service-Oriented Architecture (SOA) into IoT. It reuses web standards and principles including Hypertext Transfer Protocol (HTTP), Uniform Resource Identifier (URI), Extensible Markup Language (XML), Representational State Transfer (REST), etc. to organize the systems and physical objects with a 'mashup' architecture. This organization benefits the end-users as they are able to retrieve data and control systems in a unified way (e.g., web services), without concerning the internal implementation and transformation. Thus, WoT is widely used in various IoT applications, such as smart buildings [3] and smart shops [4].

However, the security of WoT faces a great challenge because of several inherent characteristics. Generally, the smart objects and devices might be deployed in *security-concerned and privacy-concerned environments*, such as product tracking, vehicle systems, e-health infrastructures, and household appliances. From the perspective of an enterprise, WoT is a promising technology in the improvement of their management of product flow, distribution, and storage [5]. On the other hand, a trade secret is usually involved in these transactions, which causes confidentiality and security concerns. WoT is also involved in individual users' daily life; so, it is not only protection of privacy, but also high reliability of crucial systems, such as e-health, that must be guaranteed.

As smart objects are connected with the Internet, security risks and attacks on the Web can be easily transferred to the WoT systems [6,7]. Also, the objects in WoT such as sensors

and radio frequency identification device (RFID) tags, which usually possess *limited capability* and demand extra resources to support their execution, are vulnerable to DoS attacks [8]. Last but not least, most existing security schemes, such as encryption, authentication, and security protocols, are difficult to implement in these resource-constrained objects.

Some approaches have been introduced to enhance the security and privacy of the IoT infrastructure, including RFID [9,10] and wireless sensor network (WSN) [11–14]. Meanwhile, some researchers strive to design context-aware models, frameworks, and middleware for pervasive computing (a.k.a ubiquitous computing) such as Cerberus [15], CA-RBAC [16], and UbiCOSM [17]. However, IoT conceptually differs from pervasive computing in terms of involving smart objects and the Internet infrastructure.

WoT is inherently closely connected to its context. Nearly all applications in WoT are designed to be context-aware [18,19]. Obviously, context information, including environment context, temporal context, user context, and system context, must be considered in the security and privacy issues of WoT. So far, several mechanisms have been employed to secure IoT, such as encryption [20–22], privacy protection [23–25], and access control [26,27]. However, to the best of our knowledge, there is a lack of existing work that combines context awareness with security and privacy protection in the IoT, especially in the WoT.

Context-aware security seems to be a promising vision yet a challenging task because of the nature of the contexts. First, a remarkably wide variety of properties, from temporal and spatial conditions, system and hardware status, to even human physiological states, could be considered as contexts; hence, a model supporting various context types is required. Second, context is usually presented in human-understandable terminologies: Alice's house, Bob's work time, and Eve is drunk, for instance. Thus, a security scheme should not only define the context formally in the security model, but also facilitate a user to define his/her context types and to specify his/her context-aware policies. Third, context is inherently dynamic in nature, which means a previously satisfied policy may be violated as the context could change at any time.

To address the above challenges effectively, we propose a context-aware usage control model (ConUCON) for the WoT which is based on the previously proposed a usage control (UCON) model [28]. ConUCON combines context awareness and the UCON model to introduce context information evaluation into usage control decision. It is able to provide reinforced data and resource protection regarding context information. ConUCON formally defines, describes, and supports the aforementioned wide variety of context types, from temporal and spatial conditions, system and hardware status, to human physiological states (such as body temperature and heart rate). ConUCON enables context-type definition and convenient context-aware policy specification.

On the basis of ConUCON, we design and implement a context-aware usage control framework for our ongoing SmartHome project at its middleware level, to provide data, resource, and service protection. Finally, we apply ConUCON in a remote appliance management prototype, as a case study, to demonstrate that ConUCON is able to augment security and privacy protection for the WoT.

We make the following main contributions in this paper.

- On the basis of the UCON model, we present the ConUCON model to support context-aware data, resource, and service protection for WoT. ConUCON enhances expressivity and practical usability compared with the UCON model. In addition, existing models, mostly stemming from Role-based Access Control (RBAC), are encompassed in ConUCON. This model also enables data and resource owners to employ more fine-grained security mechanisms than existing models to enhance security and privacy protection.
- ConUCON supports a wide variety of context types and provides a convenient way to define context types and specify context-aware policies. Moreover, ConUCON is a continuous usage control model. The usage decisions are not only performed prior to the access but also during the time when access takes place. In contrast, continuous usage decision is rarely supported by existing RBAC-based access control models.
- We implement a context-aware usage control framework according to the ConUCON model and deploy it on a real WoT project. The framework allows the administrator to express his policies on the data, resources, and services exposed to users.
- Finally, we demonstrate, through a working prototype, that the proposed context-aware usage control mechanism can be easily adapted to accommodate the variable requirements of security and privacy protection in a real WoT environment.

The rest of this paper is organized as follows: Section 2 describes the preliminaries and the background, including the WoT, the UCON model, as well as our SmartHome project. Section 3 describes two motivating examples. Section 4 presents the ConUCON model formally. Section 5 describes the framework based on the ConUCON model. Section 6 discusses a case study to show the application of ConUCON in a SmartHome application. Related work is introduced in Section 7, and finally, Section 8 concludes this paper.

2. PRELIMINARY AND BACKGROUND

2.1. Web of Things

The IoT is a vision of a world where all kinds of heterogeneous smart objects and devices are uniformly networked together through the Internet Protocol (IP). All the IP-enabled heterogeneous smart objects in the realm of the IoT are expected to eventually 'speak the same language', but the status quo is that hundreds of communication

protocols above the IP, such as Advanced Message Queuing Protocol (AMQP), ModBus, and Megaco, have been created by different academic groups, industrial organizations, and enterprises for their own benefit. So far, none of these proprietary protocols are widely accepted as an industrial standard. IoT is thus becoming a set of isolated networks of smart things, as few developers could grasp the whole spectrum of technologies related to these ad hoc systems.

In order to eliminate the tight coupling among these isolated networks, WoT starts to take shape while leveraging the existing ubiquitous Web protocols and standards. Compared to those aforementioned proprietary protocols, Web protocols and standards, such as HTTP, URI, and HTML, are more open, flexible, scalable, and widely accepted. Thus, integrating the smart objects and devices into the Web is a more practical and promising direction.

The WoT implementations introduce SOA as an enabler of interoperability among distributed smart objects and devices. They leverage SOAP and Web Service Definition Language (WSDL), as well as RESTful Web Services [29], to provide WS-* Web service [30] in a simple and lightweight manner [2].

2.2. The UCON model

The usage control model (UCON) [31–34] is a generalized security model proposed by Sandhu *et. al* to cover a variety of security aspects including obligations, conditions, continuity, and mutability. As depicted in Figure 1(a), the UCON model consists of eight components: subjects, subject attributes, objects, object attributes, rights, authorizations, obligations, and conditions. The first five hold similar meaning with the concepts in traditional access control models, whereas authorizations, obligations, and conditions impact the usage decision. Authorizations permit or deny access from a subject to an object with a particular right on the basis of the attributes of the subject and object. Obligations require the subject to perform specific actions before (pre) or during (ongoing) an access. Conditions are environmental factors.

The UCON model is further extended on the basis of attributes mutability, which allows updates to the attributes. Attributes can be immutable or updated before (pre), during (ongoing), and after (post) the usage. Detailed descriptions can be found in [32,33].

The UCON model is proposed to involve comprehensive aspects related to usage control. Context properties are actually contained in the condition (C) component. Nevertheless, the model is inherently generic in terms of context expression. Thus, we believe it is necessary to refine the model to enhance its expressivity and usability. In our previous work, we have combined context awareness with the UCON model to enhance privacy protection and resource usage control in the Android platform [35]. The work demonstrates that the context-aware UCON is promising for the smart phone platforms. Thus, we are inspired to adopt context-aware UCON in WoT, where the scenario is different from the smart phone platform in several aspects such as participants, context types, and protection objectives.

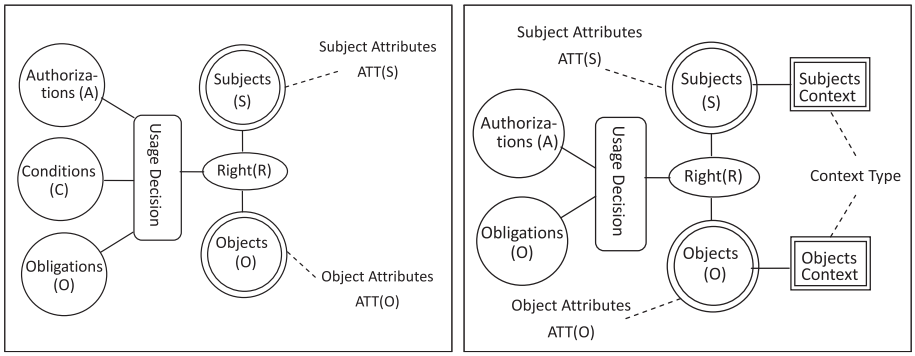
2.3. Smarthome project

The SmartHome project is one of our ongoing projects, which aims to construct a home automation platform. It is designed to support a variety of applications, including remote household appliances control, home security and protection, home energy management, etc. It organizes a large spectrum of smart devices such as sensors, RFID readers and tags, magnetic contacts, smart sockets, and dimmers, which are equipped in the house. Also, it manages all the devices and their communication, hides the hardware and protocol disparities, as well as provides a uniform service interface to the application. By wrapping their functionality into RESTful Web Services, SmartHome integrates these heterogeneous smart devices in a WoT style.

As depicted in Figure 2, the architecture can be partitioned into the following layers: device layer, communication network layer, and middleware layer.

2.3.1. Device layer.

The system integrates multiple smart devices, including wireless sensors that measure temperature, lighting, as well as humidity, RFID readers and tags, magnetic



(a) The Usage Control Model [51] (b) ConUCON: a Context-Aware Usage Control Model

Figure 1. The UCON Model and the ConUCON Model. (a) The usage control model and (b) ConUCON: a context-aware usage model.

DOI: 10.1002/sec

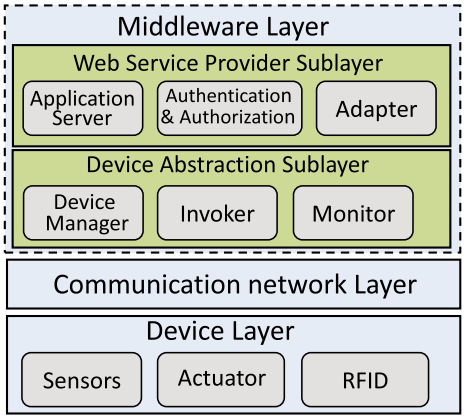


Figure 2. The architecture of the SmartHome project.

contacts, smart sockets, and dimmers. These heterogenous devices can be classified into two main categories: sensors and actuators.

2.3.2. Communication network layer.

Smart devices communicate with our system via a variety of protocols and standards, such as ModBus, Devices Profile for Web Services (DPWS), and OPC-UA, which forms several isolated communication networks.

2.3.3. Middleware layer.

The middleware layer aims to hide the heterogeneity among smart devices and offer RESTful Web Services to IoT users and application developers.

When connected to the system for the first time, new devices should register its meta-data to the *device manager*. The *monitor* will track the status of devices during runtime, and the *invoker* will take the responsibility of translating bidirectional communication. The above three components form a sublayer called *device abstraction*.

On the basis of the device abstraction, the Middleware offers a *Web service provider*, which contains the *application server*, the *adapter*, and the *authentication and authorization* component. The *application server* provides runtime support for Web Services that are created by the adapter through orchestrating functionalities and data from devices below or just wrapping them up. The authentication and authorization component carries out the authentication and authorization with mechanisms such as password, access control, and usage control. The Web services can only be invoked by applications after completing authentication and authorization.

3. MOTIVATING EXAMPLES

This section describes two motivating examples related to privacy protection and service protection in our SmartHome applications.

3.0.0.1. Confidentiality and privacy protection. An e-health system in the SmartHome collects diagnosing-used of physiological data from one of the family members. These data are stored in the database of the SmartHome. The physician uses a web service to access the data through the Web. Without extra protection, it might result in a privacy exposure. To prevent this, the family member can assign a constraint on the data to ensure that the data are only readable in the diagnostic room by the specified physician.

3.0.0.2. Resource and service protection. To enable the family members to control the home appliances when outside the house, a remote appliance management system in the SmartHome encapsulates the control functions into Web services. By invoking them, a family member can remotely control his air-conditioner using his cell phone. However, to save energy consumption, the family member could assign the air-conditioner a usage constraint, such that the air-conditioner will be powered up only when indoor temperature is higher than 26°.

4. CONUCON: A CONTEXT-AWARE USAGE CONTROL MODEL

The UCON model, summarized in Section 2.2, is a promising security model suitable for modern information security requirements. By extending the UCON model, we present a context-aware usage control model ConUCON, which leverages context awareness to provide reinforced data and resource protection. ConUCON extends the UCON model to accommodate itself in the WoT, because the UCON is designed to cover nearly all aspects in access control. ConUCON serves as the foundation of our usage control framework for the SmartHome project.

ConUCON consists of three major parts: model components, administrator policy specifications, and runtime usage decisions. The model components are able to represent all factors in WoT, and the administrator policy specifications enable an administrator to specify his policies on the services in WoT, whereas the runtime usage decisions perform usage decision at runtime.

4.1. ConUCON model components

As presented in Figure 1(b), ConUCON contains the following components: subjects, objects, states (which involve subject attributes, object attributes, and their values), rights, authorizations, obligations, contexts (which include subject context and object context), and context types. The formal definitions and essences of these components are presented in this section.

Similar to most access control models, the concepts of subjects and objects are the entities that initiate access behavior and entities that are passively accessed, respectively.

Definition 1. ***Subjects*** *and* ***Objects****. A subject is an entity that holds and exercises certain rights on objects. An object is an entity that the subjects can access or use. The subject set* and *object set* are denoted by *S and O*, *respectively.*

Example 1. In the WoT scenario, subjects are the applications and users that attempt to access services, retrieve data, etc., whereas objects are all of the entities that are accessed or used in the WoT, including services, resources, and devices.

Definition 2. ***Attributes***. *An attribute is a property used in usage decisions, such as UIDs, invoking the frequency, path of a service, confidentiality and integrity level, etc. All attributes comprise the attribute set* (*AT*).

Each subject and object is associated with a corresponding attribute set to record and maintain the properties related to usage decisions. The attribute set can be queried using the following function:

$$\tau : S \cup O \rightarrow \mathcal{P}(AT) \tag{1}$$

For a subject or object, $so \in S \cup O$, which holds an attribute $at \in \tau(so)$. The value of the attribute $so.at$ can be retrieved with the function

$$\upsilon : (S \cup O) \times \tau(S \cup O) \rightarrow ran(\tau(S \cup O)) \tag{2}$$

where *ran(a)* is the value of attribute *a*.

Example 2. Some resource-hungry objects in the WoT, such as antennas that consume significant battery energy to transmit signals, are vulnerable to DoS attacks. To protect such devices, administrators may limit the usage frequency by an individual application within a certain range. In this scenario, the usage history should be accounted as an attribute of the device, which will be evaluated during the usage decision process.

Definition 3. ***State***. *A state is defined as a set, whose elements are triples* (so, at, val), *where*

$$so \in S \cup O,\ at \in \tau(so), \text{ and } val = \upsilon(so, at).$$

A state element contains several attributes that are involved in one usage decision process, as well the owner of the attributes and corresponding attribute values. A state is a subset of the *state set* (*ST*) that contains all the attributes and their values.

The state is variable during and after the usage, which is achieved by an *update action*:

$$\mu : \mathcal{P}(ST) \rightarrow \mathcal{P}(ST). \tag{3}$$

Definition 4. ***Rights***. *A right is an operation that subjects can perform on objects. All rights comprise the right set* (*R*).

In WoT, rights can be divided into several functional categories according to the object types. For files and other data, the rights are *read*, *write*, *delete*, etc., whereas for resources and services, the rights include *use*, *invoke*, *disable*, etc.

The right set is defined by users, who can also define partial rights according to Jaehong and Ravi [31]. For example, the user can define partially read and modify, that is $R = \{Read, Mod, \alpha\}$, where *Read* denotes the read action, *Mod* denotes the modification action, and α denotes a partial range. α constrains read and modify in a range of $0 \leq \alpha \leq 1$. Through this way, rights can be performed on a part of a file but not all of it. This fine-grained manner is especially practical in WoT.

Example 3. As mentioned in the motivating example in Section 3, through monitoring, an e-health system gathers miscellaneous data from a patient. For the protection of the patient's privacy, surgery data are only available for his surgeon, heart data only for his cardiologist, and so on. Thus, partially reading the physiological data will meet this requirement.

In WoT, services are shared between collaborating partners. In adapting to this scene, ConUCON employs the permission label model to represent the security levels of subjects and objects, examples include Decentralized Information Flow Model (DIFM) [36] and the Android security model [37].

Definition 5. ***Permission Labels***. *A permission label is a credential that allows a subject to perform a specific right on corresponding objects. Permission labels are assigned to subjects and objects. All permission labels comprise the permission label set* (*P*).

For a subject, its permission labels determine which objects it can access, whereas the labels for an object determine which subjects can access it. Each subject owns a permission label set, which can be retrieved using the function

$$\varphi_s : S \rightarrow \mathcal{P}(P) \tag{4}$$

Each resource object and service object can be attached with a permission label to declare the permission required to use it. The function

$$\varphi_o : O \rightarrow P \tag{5}$$

is defined to query the label.

It is a bit more complex for data objects. Each of the objects has two labels, one for confidentiality and the other for integrity. The confidentiality label is an element of the *confidentiality label set (*CL*)*, whereas all integrity labels comprise the *integrity label set (*IL*)*. A subject is also associated with these two labels to indicate its confidentiality level and integrity level, respectively. The orders of the confidentiality level and integrity level are denoted by $\{\leq_c, \geq_c\}$ and $\{\leq_i, \geq_i\}$, respectively.

The functions

$$\varphi_c : S \cup O \rightarrow CL \tag{6}$$

$$\varphi_i : S \cup O \rightarrow IL \tag{7}$$

are defined to retrieve the confidentiality and integrity labels of a data object or a subject, respectively.

Definition 6. ***Obligations***. *An* **obligation** *is a mandatory action that must be performed before or during an access. It is an element of the obligation set (OB).*

Example 4. To allow a user to remotely manage the power supply of the house, we made the functionality of powering off an electricity supply unit wrapped as a web service and exposed it on the Web. Meanwhile, the unit supplies power for several appliances including computers. To avoid damages on the appliances resulting from sudden power failure, the user must turn off other web services before turning on the electricity supply unit.

4.2. Context

Context awareness has been proposed for more than a decade. Recently, it has become more popular because of the rise of pervasive computing, mobile computing, and IoT. Many works have presented the definitions of context, among which, the one proposed by Chen *et al*. [38] is widely accepted. They regard *context* as a set of environmental states and settings that determine an application's behavior, including a broad range from system state to network status, from human physiological conditions to human emotions, and from time to the physical world. To accurately and appropriately describe the WoT, we focus on the contexts related with the environment, time, human, and system.

We define context in ConUCON as follows.

Definition 7. ***Contexts***. *A context is defined as a property of the environment, time, human, and system, which is related to security and privacy protection. This property is called the context type*, which is an element of the context type set *(CT).*

There is a subtle difference between the context and the attribute: a context is a property of the physical environment, time, human, and system, whereas an attribute is a property directly related to a subject or an object.

We divide the context into four categories:

Environment Context location of user and devices, temperature, light intensity, humidity, etc.

Temporal Context temporal information.

Human Context the user's profile, people nearby, user's body temperature, etc.

System Context the property of the system, such as battery, network bandwidth, etc.

4.2.1. Environment context.

An *environment context* is a property of the environment where a subject or an object settles in. It includes *spatial context* and *physical context*. The physical context is a set of properties of the physical environment, such as temperature, light intensity, humidity, etc., which are measurable and can be easily evaluated during usage decision. However, the spatial context is more complicated to evaluate. Thus, this section presents a detailed definition of the spatial context.

A *spatial context* is defined as a spatial property. Spatial context $\in$ CT. We adopt the geometric model of GEO-RBAC [39] to the model the positions.

Definition 8. ***Feature and feature type***. *A feature is an object that indicates an entity that occupies a space in the real world, which is identified by a feature name. The features are included in the feature set* (*F*). *Each feature has a feature type contained in the feature type set (FT).*

A feature can be mapped to a *geometry* on the Earth. A geometry is an object in the Euclidean space with a coordinate, which is an element in the *geometry set* (GEO).

The functions

$$\gamma : F \rightarrow FT \tag{8}$$

and

$$\xi : F \rightarrow GEO \tag{9}$$

are used to obtain the feature type and the geometry of a feature.

Definition 9. *Feature order and feature type order*
feature type order $(\leq_{ft}) : ft_1 \leq_{ft} ft_2$ *iff* $\forall f_1 \in F \wedge \gamma(f_1) = ft_1, \exists f_2 \in F \wedge \gamma(f_2) = ft_2, \xi(f_1) \subseteq \xi(f_2)$
feature order $(\leq_f) : f_1 \leq_f f_2$ *iff* $\gamma(f_1) \leq_{ft} \gamma(f_2) \wedge \xi(f_1) \subseteq \xi(f_2)$

Example 5. Office 2E315, Pentagon, Arlington, and Virginia are examples of features, whose feature types are Room, Building, County, and State, respectively. The Room and Building satisfy the $\leq_{ft}$ order, whereas Arlington and Virginia satisfy the $\leq_f$ order.

Definition 10. *(Real Position and Logical Position)* A *real position is a position on the Earth and can be obtained using a device, such as a GPS-based equipment, whereas a logical position is a semantic representation of a position. The* **r***eal position set and logical position set are denoted as RP and LP, respectively.*

Obviously, a real position corresponds to a geometry, and a logical position corresponds to a feature. A real position may correspond to one or more logical positions under different feature types. For example, a region may

correspond to a room or part of a city when assigning it with these two feature types.

The function

$$\rho_{ft} : RP \rightarrow LP \tag{10}$$

is used to map out a real position to the corresponding logical position under the feature type *ft*.

Thus, we can define the inclusion relation between a real position and a logical position $\sqsubseteq_p$: $rp \sqsubseteq_p lp$, where $rp \in RP \wedge lp \in LP$ *iff* $\rho_{\gamma(lp)}(rp) \leq_f lp$.

4.2.2. Temporal context.

A temporal context is defined as a temporal property. *Temporal context* $\in$ CT.

Definition 11. ***Time Instants***. *A time instant is a time point that has the form specified in ISO 8601 [40]:*

$$TI := YYYY - MM - DDThh : mm : ss$$

where

$$MM \in \{1, 2, \ldots, 12\} \wedge DD \in \{1, 2, \ldots, 31\} \wedge YYYY \in \mathbb{N} \wedge hh \in \{0, 1, \ldots, 23\} \wedge mm, ss \in \{0, 1, \ldots, 59\}$$

The definition of the periodic expression in ConUCON is based on past studies in [41]:

Definition 12. ***Periodic Expression***. *The periodic expression is defined as*

$$\begin{aligned} PE &:= Y|W \\ Y &:= R.years|R.years \triangleright S.years|R.years + M \\ W &:= weeks|weeks + D \\ M &:= R.months|R.months \triangleright S.months \\ &\quad |R.months + D \\ D &:= R.days|R.days \triangleright S.days|R.days + H \\ H &:= R.hours|R.hours \triangleright S.hours|R.hours + M \\ M &:= R.minutes|R.minutes \triangleright S.minutes \end{aligned}$$

where

$$R \in 2^{\mathbb{N}} \cup \{all\}, S \in \mathbb{N}$$

Example 6. We can use the periodic expression years + 7. months $\triangleleft$ 6.months to indicate the second half of every year and the expression weeks + {1, 2, . . .5}Days + 9.hours $\triangleleft$ 8. hours to indicate the working hours of every week.

As a result, we can define the inclusion relation between a time instance and a periodic time [41], $\sqsubseteq_t : ti \sqsubseteq_t <$ [begin, end], $P >$, if and only if there exists a time interval $it \in \Pi(P)$, such that $ti \in it$ and begin $\leq ti \leq$ end, where $<$ [begin, end], $P >$ is a periodic time. *Begin* and *end* are two time instants; $\Pi(P)$ is the set of time intervals corresponding to the periodic expression *P*.

Example 7. The periodic expression $PT =<$ [2011−01−01*T*00 : 00 : 00, 2012 − 12 − 31*T*23 : 59 : 59], weeks + {1, 2, . . . 5}.Days + 9.hours$\triangleright$8.hours $>$ indicates the working hours during the years 2011 and 2012 with a time instant 2011-04-19 T14:30:00 $\sqsubseteq_t$ PT.

4.2.3. Human context.

A human context is defined as the human's physiological condition. Human context $\in$ CT.

On one hand, human centric service is a key objective of WoT. On the other hand, thanks to the miniaturization of electrical devices, it is now possible to collect real-time information on human's physical condition with the use of the wearable systems [42]. Thus, some works [43,44] combine the human physical condition with context awareness, which is called human context.

Example 8. Once a wearable system or vehicle system detects that the amount of alcohol in a driver's blood is over the legal limit, it should stop that individual from driving.

Human context contains *measurable* conditions such as body temperature, heart rate, etc., as well as the *inferrable* state, such as the emotion and the drunken state in Example 8.

4.2.4. System context.

A *system context* is defined as the state and property of the system in WoT. System context $\in$ CT.

In WoT, the system relies on both the components and the communication networks to run. Thus, the system context is related to the resource conditions in the components and network status. Thus, the system context includes component context and network context: The *component context* is the state of the components, such as battery, CPU utilization, etc. The *network context* reflects the condition of the communication networks which the system relies on, such as session count, network bandwidth, etc.

Obviously, an element of physical context, human context, and system context such as temperature, battery, heart rate, and so forth, could be easily denoted as a *tag*. Thus, we define context properties as corresponding tag Boolean expressions.

Definition 13. ***Context Property***. *A context property is a semantic phrase that describes a context state. A context property is satisfied if the corresponding tag Boolean expression (TagConstraint) is true, where*

$$TagConstraint ::=< tagPredict > relator < value >$$

tagPredict is an operation expression that uses the *tags* and number as operands, and *relator* is a logical operator.

DOI: 10.1002/sec

Example 9. A context property *lowTemperature* could be bound with a tag expression

$$\text{temperature} \leq 20^{\circ}C$$

4.3. Administrator policy specification

To allow an administrator to specify his context-aware security policy, ConUCON defines the policy specification. A policy in ConUCON describes the following functionalities:

- Which permission label should be assigned to a service object? Which confidentiality label and integrity label should be assigned to a data object? Which permission label set (including confidentiality labels and integrity labels) should be assigned to a subject?
- If a subject requests to perform a specific action (right) on an object, what authorizations, obligations and context constraints should be satisfied before (pre) and during (ongoing) the access?

Definition 14. ***Label Policies***. *A label policy is used to specify the permission label that will be assigned to a service object, the confidentiality label and integrity label assigned to a data object, or the permission label set (including confidentiality labels and integrity labels) assigned to a subject.*

We define a function

$$\varpi_{o_1} : O \rightarrow P \tag{11}$$

to enforce a permission label to a resource object, a function

$$\varpi_{o_2} : O \rightarrow CL \times IL \tag{12}$$

to enforce confidentiality label and integrity label to a data object, and a function

$$\varpi_s : S \rightarrow \mathcal{P}(P) \tag{13}$$

to grant a permission label set to a subject.

Definition 15. ***Usage Control Policy***. *A usage control policy is used to specify authorizations, obligations, and contexts that should be satisfied before (pre) and during (ongoing) a subject's performance of a specific action (right) on an object. All the usage control policies are included in the usage control policy set (UP).*

$$UP \subseteq S \times O \times R \times PreOb \times OnOb \times StateConstraint \times PreContext \times OnContext \times Update$$

where

- $PreOb, OnOb \in P(OB)$
- $StateConstraint ::= (StatePredicate) \mid \neg StateConstraint \mid StateConstraint \vee StateConstraint \mid StateConstraint \wedge StateConstraint.$ StatePredicate is a relational expression in the form of $f(P(S \cup O \times AT))relator < value >$, where f is an operational expression that uses attributes as operands, and the *relator* is a logical operator.
- $PreContext, OnContext \subseteq ContextConstraint$, $ContextConstraint ::= (ContextPredicate) \mid \neg ContextConstraint \mid ContextConstraint \vee ContextConstraint \mid ContextConstraint \wedge ContextConstraint$ $ContextPredicate ::= TagConstraint \mid PeriodicTime \mid LP$
- $Update ::= \mu$; μ is defined as Function 3.

Example 10. Let us consider the *service protection* example in Section 3 to illustrate the administrator policy specification. To adjust the indoor temperature in the house to a comfortable temperature, he or she would turn on the air-conditioner remotely with the cell phone when he or she is on the way home. For the sake of device protection and energy saving, the administrator's constraints on invoking the power through web service are as follows:

- The air-conditioner will not be turned on during the family member's work time.
- The family member should be in his vehicle or office when he uses his cell phone to turn on the air-conditioner. With this policy, if he loses his cell phone, the one who acquires the phone is not able to control the appliances.
- Before turning on the air-conditioner, the user must invoke another web service to close the windows and ventilation fans.
- When the control system receives the power on command, it will not power on the air-conditioner if the indoor temperature is below 26°C.
- The windows and ventilation fans should not be opened while the air-conditioner is working.
- The air-conditioner will power off when the indoor temperature is below 26°C.
- If the family member does not arrive home 30 min after turning on the air-conditioner, the system will turn it off.

We can specify the above policies just as in Table I.

4.4. Runtime usage decisions

We employ the Bell–LaPadula model [45] for confidentiality and the Biba model [46] for integrity to express the authorizations in the ConUCON. Of course, other appropriate security models can also be used to express specific application constraints in the ConUCON if needed.

Definition 16. ***Authorizations***. *Authorizations are used to check whether a subject is allowed to perform an action on an object, according to a specified security model such as the integrity models and confidentiality models.*

Table I. An example of a usage control policy.

Components	Constraints
Subject	All
Object	WebServiceID_1 (predefined as starting the air-conditioner)
Right	Invoke
Pre-obligation	ObligationID_1 (predefined as closing or reminding the user to close the windows and ventilation fans)
On-obligation	ObligationID_2 (predefined as keeping windows and ventilation fans closed)
State	$RFID.in \vee (currentTime - startTime \leq 30 minute)$
Precontext	$(subject.vehicle \vee subject.office) \wedge (\neg < [2011-01-01T00:00:00, 2011-12-31T23:59:59], weeks + \{1, 2, \ldots 5\}.day + 9hours \lhd 8hours >)$
Ongoing context	$indoor\ temperature \geq 26^{\circ}C$
Update	$if(onStart) then\ startTime := currentTime$

RFID, radio frequency identification devices.

The function used to obtain the authorization

$$\Omega : S \times O \times R \rightarrow \{\text{true},\ \text{false}\} \tag{14}$$

is defined in Figure 3.

Definition 17. ***Usage Decision***. The *usage decision* determines whether an access should be permitted or an ongoing access should be revoked on the basis of authorizations, obligations, contexts, and states. The usage decision is performed as:

- $allow\ (s, o, r) \Rightarrow \Omega(s, o, r) \wedge fulfill\ (preOb) \wedge fulfill(preContext) \wedge fulfill(stateConstraint)$
- $revoke\ (s, o, r) \Leftarrow \neg fulfill(onOb) \vee \neg fulfill(onContext) \vee \neg fulfill(stateConstraint)$
- $update\ (state)$

The revoke (s, o, r) function reflects the concept of *continuous evaluation*. Continuous evaluation is critical in the WoT, because its participants are usually on the move.

Example 11. As described in the motivating example in Section 3, the private physiological data are restricted to be read only in the hospital. A doctor has acquired the permission to read the data. Afterwards, while browsing the data, the doctor roams out of the restricted area unconsciously. The system should trigger a warning or even take actions such as physically deleting the data as soon as it detects this situation.

Thus, in the ConUCON, the evaluation of context constraints is performed both before (pre) and during (ongoing) usage.

5. A CONTEXT-AWARE USAGE CONTROL FRAMEWORK FOR WEB OF THINGS

On the basis of the above ConUCON model, we developed a context-aware usage control framework for WoT.

5.1. Framework overview

Figure 4 describes the architecture of the framework where objects can communicate with each other by using the messaging mechanism listed in Table II.

The circled numbers in the figure indicate the processing flow during one usage decision process. When an application or a user tries to access an object or a functionality provided as a web service, the policy enforcement point (PEP) perceives the request and invokes the policy decision point (PDP) by using $request(s, o, r)$. Then, the PDP performs the authorization and activates the policy information point (PIP) with $evaluate(s, o, r)$. The PIP then invokes the policy resolver to resolve predefined policies related to s and o and then invokes the evaluation engines to check the prepolicies. After that, the PIP sends a result (i.e., a $fulfill(s, o, r)$ or a $violate(s, o, r)$ message) to the PDP, which synthesizes the received result and the authorization result, to decide whether the access should be permitted or denied and notifies the PEP of the decision by using a $permit(s, o, r)$ or a $deny(s, o, r)$ message. The state evaluation engine also updates the states accordingly.

5.2. Continuous evaluation

As mentioned in Example 11, continuous evaluation is critical in the WoT. We implement it in the ConUCON.

$$\Omega(s,\ o,\ r) \Rightarrow \begin{cases} \varphi_o(o) \in \varphi_s(s), \text{if } o \text{ is a resource object} \\ \varphi_c(s) \geqslant_c \varphi_c(o) \wedge \varphi_i(s) \leqslant_c \varphi_i(o), \text{if } o \text{ is a data object, and } r = \text{read} \\ \varphi_c(s) \leqslant_c \varphi_c(o) \wedge \varphi_i(s) \geqslant_c \varphi_i(o), \text{if } o \text{ is a data object, and } r = \text{write} \end{cases}$$

Figure 3. The authorization function.

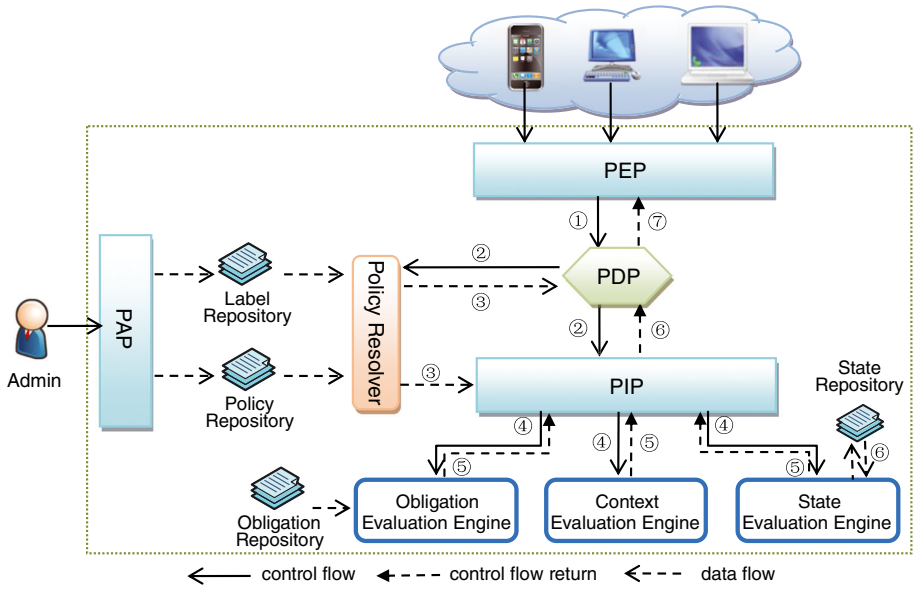


Figure 4. The ConUCON framework.

After allowing access, the evaluation engines begin to evaluate ongoing policies continuously. Once a violation is detected, the engines notify the PIP, which then sends a *violate*(s, o, r) message to the PDP, and the PDP will send a *revoke*(s, o, r) message to the PEP immediately to revoke the access. That is, the PIP can send a *violate*(s, o, r) message to the PDP once the engines detect a violation, whether the PDP sends an *evaluate*(s, o, r) message or not.

Continuous evaluation is terminated when an application stops accessing an object. Under this circumstance, the PEP notifies the PDP through a *terminate*(s, o, r) message, which then sends a *withdraw*(s, o, r) message to the PIP to withdraw the continuous evaluation in this session.

Continuous evaluation is an important improvement in the existing access control work, whose usage decision only occurs before the access. In particular, the web service that is enforced with an ongoing policy should be implemented as stateful to enable the PEP to maintain the session.

5.3. Framework components

5.3.0.1. Policy enforcement point. The PEP is responsible of perceiving access request and termination, invoking the PDP to perform usage decisions and enforcing usage control according to the PDP's response. When the PEP captures a request, it invokes the PDP with a *request*(s, o, r) message. The PEP allows access only if the PDP returns a *permit*(s, o, r) message. After permitting access, the PEP shifts to a listening state. If any of the ongoing policies is violated, the PEP will be notified by the PDP with a *revoke*(s, o, r) message to terminates the access. In addition, the PEP should perceive the termination of access. Once a subject terminates the access, the PEP sends a *terminate*(s, o, r) message to the PDP, and then the PDP stops monitoring policies.

5.3.0.2. Policy decision point. The PDP is the component that performs usage decisions. It is responsible for activating the policy resolver and the PIP, authorizing

Table II. Message types transmitted among the components.

Message	Source	Destination	Meaning
request(s, o, r)	PEP	PDP	The subject s is requesting to perform the right r on the object o.
permit(s, o, r)	PDP	PEP	The *request*(s, o, r) is permitted.
deny(s, o, r)	PDP	PEP	The *request*(s, o, r) is denied.
terminate(s, o, r)	PEP	PDP	The subject s terminates access to the subject o.
revoke(s, o, r)	PDP	PEP	Revoke the *request*(s, o, r).
evaluate(s, o, r)	PDP	PIP	Perform obligation, state, and context evaluation.
fulfill(s, o, r)	PIP	PDP	The obligation, state, and context policies are enforced.
violate(s, o, r)	PIP	PDP	Not all obligations, state, and context policies are enforced.
withdraw(s, o, r)	PDP	PIP	Withdraw all continuous evaluations.

PEP, policy enforcement point.
PDP, policy decision point.
PIP, policy information point.

(i.e. checking the permission labels), and notifying the PEP of the usage decision result after merging the authorization result and the responding result of the PIP. The PDP is invoked by the PEP when access actions, including request and termination, occur.

When the PEP captures an access request, it invokes the PDP with a *request*(s, o, r) message. The PDP then retrieves the permission labels of s and o from the label repository and performs authorization on the basis of Definition 16. If the result is true, the PDP then invokes the PIP to gather information related to the usage decision (i.e. prepolicy evaluation result). If any policy is violated, the PDP returns a *deny*(s, o, r) message to the PEP to deny access. Otherwise, it returns a *permit*(s, o, r) message to the PEP and listens for both the PEP and the PIP to process the *violate*(s, o, r) from the PIP and *terminate*(s, o, r) from the PEP.

5.3.0.3. Policy information point. The PIP is the component that provides the PDP with the evaluation information on obligation, state, and on text both before (pre) and during (ongoing) access, with the aid of the obligation evaluation engine, the state evaluation engine and the context evaluation engine.

The PIP is invoked by the PDP with an *evaluate*(s, o, r) message. The PIP then calls the policy resolver to resolve policies that contain prepolicies and ongoing policies. After that, the PIP invokes the evaluation engines to evaluate prepolicies first. If any of the engines return a false result, the PIP returns a *violate*(s, o, r) message to the PDP. Otherwise, the PIP returns a *fulfill*(s, o, r) message then invokes evaluation engines to fork daemons to evaluate ongoing polices continuously. If any of the ongoing policies are violated, the PIP notifies the PDP with a *violate*(s, o, r) message. The PIP also listens on the PDP after the prepolicies evaluation. When it receives a *withdraw*(s, o, r) message from the PDP, it withdraws the ongoing evaluation.

5.3.0.4. Evaluation engines. The evaluation engines are invoked by the PIP to perform a corresponding policy evaluation for the obligations, states, and contexts.

The obligation evaluation engine monitors the execution of obligations. If the obligation is an action that can be carried out by the engine directly, the engine can require the subject to perform the obligation or carry it out directly. Recall Example 10. The engine can remind the user to call another web service to close the windows and ventilation fans or just call the corresponding web services directly. If the obligation can only be observed, the engine does not return a true result until the obligation is observed. The obligations defined by Definition 6 are stored in the obligation repository, which can be accessed using their obligation IDs or paths specified by the user.

The state evaluation engine is invoked to evaluate the state constraints. It first resolves the attribute type from the state constraint expressions and retrieves the corresponding attribute values from the state repository. Then, it evaluates whether the constraint is satisfied and notifies the PIP of the evaluation result. Besides, the state evaluation engine will update the state values into the state repository if needed.

The context evaluation engine evaluates the context policies and monitors the change of the context. Similar to the state evaluation engine, it first resolves the context types from context constraint expressions. Then, it interacts with underlying systems and sensors to retrieve context value such as the coordinates, CPU utilization, and battery power information.

5.3.0.5. Policy administration point. The policy administration point (PAP) is a component that interacts with the user, which allows the user to administrate the usage policies for the data and resources in his smart phone. The user can impose or deprive the permission labels, confidentiality labels, and integrity labels to a subject or an object as in Definition 5.

The PAP then formats the user's policy specification and stores the policies into the label repository and policy repository, respectively. The policies are formatted in XML, which will be discussed in Section 5.4.

5.4. Policy specification

Through the PAP, the user specifies his usage policies on his data and resources according to Definition 14 and 15. In order to facilitate policy storage and transmission among the components, the policies are represented in an XML format. There are several XML-based security policy specification languages available, such as eXtensible Access Control Markup Language (XACML) [47], Security Assertion Markup Language (SAML) [48], and WS-Policy [49], which are based on the RBAC access control model. We extend them to support the ConUCON. The primary tags used in ConUCON are listed as follows.

- <Subject>, <Object> and <Right> specify the subject, object, and right associated with the policy.
- <Obligation> tag specifies an obligation. The *ObligationTime* specifies whether the obligation must be performed before (pre) or during (ongoing) the access, whereas the *ObligationID* specifies the ID of the obligation. The obligation evaluation engine can retrieve the action stored in the obligation repository with this ID. The user can specify a new obligation by assigning the action path to the *ObligationID* and use several <Parameter> tags to specify the parameters to execute the action.
- <State> tag specifies the state constraint in the policy. The <Attribute> tag indicates the attribute in this state constraint, whereas the attribute *Owner* indicates the owner of this attribute, and the *Type* represents the attribute type. The <Expression> tag specifies the logic expression expected, which is defined in Definition 15.
- <Context> tag specifies the context constraint in the policy. The meaning of *ContextTime* is similar to that of *ObligationTime*. The context consists of several <ContextComposition> tags, which are connected

with "∧". Each < ContextComposition > consists of several < Factor > tags connected with the *Operator*. The < Factor > is a context predicate defined in Definition 15; the *Type* specifies the context type defined in Definition 7.

- < Update > tag specifies an update policy. The *UpdateTime* declares the time to perform this update, which is in {Allow, Deny, Ongoing, Post}. The < Attribute > tag represents the attribute to be modified. It is stored in the state repository and identified by *Name*, whereas the default value of the attribute is *Default*. An < Expression > tag specifies an assignment expression that is executed to update the state.

5.5. Deployment

The whole framework is deployed in the middleware layer (See Figure 2) of the SmartHome system.

5.5.0.1. The PEP. The PEP is integrated in the application server to capture access requests to all web services. Whenever an application invokes the web service, it resolves the subject (i.e., the application), the object (i.e., the web service), and the permission labels from the request.

5.5.0.2. The evaluation engines. The evaluation engines are implemented as a daemon to monitor and evaluate the ongoing policies continuously. In particular, the context evaluation engine performs context data acquisition and reasoning. It collects the raw context data through the interfaces provided by the middleware and then processes and reasons [50,51] the collected data to make them meaningful for the usage decision process.

6. REMOTE APPLIANCE MANAGEMENT IN SMARTHOME: A CASE STUDY

We build a remote appliance management prototype within our SmartHome project to enable family members to remotely manipulate the household appliances. We also implement a prototype of the air-conditioner management system presented as the motivating example in Section 3 and Example 10.

As a case study, we employ the ConUCON to perform policy specification and runtime enforcement in the air-conditioner management scenario. The objective of the case study is to demonstrate how to utilize the ConUCON to enhance privacy and security protection in the WoT.

6.1. Scenario overview

In this scenario, family members can manage the air-conditioner by invoking a web service through using their cell phones. Through this method, one of the family members can turn on the air-conditioner remotely to cool the house prior to arriving home. Meanwhile, for the sake of device protection and energy saving, the family can also enforce several constraints on the remote management, as listed in Example 10 and Table I.

6.2. Prototype design

6.2.1. Components.

The smart objects and devices used are illustrated in Figure 5.

Smart socket	It provides eight socket outlets to power the appliances and electronic devices connected to it. In addition, it is accessible via web services, that is, the powering on and powering off of each outlet can be managed through theweb.
Sensors and base stations	The sensors are distributed in the house to acquire environment context data, including indoor temperature and humidity. Each of the sensors runs on the TinyOS 2.1 operating system and provides RESTful style web services.
Tags and RFID Readers	The RF-based tags are carried by the family members. When a tag that is used to identify the carrier enters the house, it will be detected by the RFID reader. The reader is connected to a PC as a gateway via the ModBus protocol. The gateway then provides RESTful style web services to the upper layer.

(a) Smart Socket (b) RFID Reader and Tags (c) Sensors and Base Station

Figure 5. Smart objects and devices in a remote appliance management system. (a) Smart socket, (b) RFID reader and tags, and (c) sensors and base station.

6.2.2. Implementation.

Our remote appliance management prototype is constructed as shown Figure 6. It offers a vision of remote appliance management leveraging the aforementioned smart objects, as well as the SmartHome system described in Section 2.3. Devices are connected to the device abstraction layer through the communication network with different protocols and standards, including the devices profile of web services (DPWS) provided by the smart sockets, as well as the RESTful style web services offered by the wireless sensor network motes and the RFID reader gateway.

The web service provider employs Java API for XML Web Services (JAX-WS) to build an application server. It provides two pairs of web services for this scenario. The first pair (mark them as *ServiceOn1* and *ServiceOff1*) is used to remotely power on and power off the first socket outlet to turn on and turn off the air-conditioner, respectively. Whereas, the second pair (mark them as *ServiceOn2* and *ServiceOff2*) is defined to remotely power on and power off the second socket outlet to control the ventilation fan.

6.3. Case analysis

Subjects and objects The subject, in this scenario, is the mobile phone application (marked as *Application GUID*) accessing the remote appliance management system. In fact, if needed, the application can be assigned a role according to the family member who uses it. The role can be one of $\{Elderly, Parent, Child, Worker\}$. The object in this scenario is the *ServiceOn1*.

States In this scenario, only the working time of the air-conditioner is involved; thus, the state contains only one triple, that is, $state = \{(ServiceOn1, startTime, \upsilon(ServiceOn1, startTime))\}$.

Rights The right set $R = \{Invoke\}$, and right $r \in R$.

Permission labels The permission label set $P = \{cn.pku.sei.serviceOn1.INVOKE, cn.pku.sei.serviceOn2.INVOKE, cn.pku.sei.serviceOff1.INVOKE\}$.

Obligations The pre-obligation (marked as *cn.pku.sei.obligation1*) is to shut down the ventilation fan or call *ServiceOff2* before invoking *ServiceOn1*. The ongoing obligation (marked as *cn.pku.sei.obligation2*) is to keep ventilation fan off while the air-conditioner is working. The evaluation engine can check whether the user enforces this obligation by checking the state of the second socket outlet.

Context Context data in this scenario include the spatial context, temporal context, and indoor temperature of the subject, that is, $CT = \{spatial\ context, temporal\ context, temperature\}$.

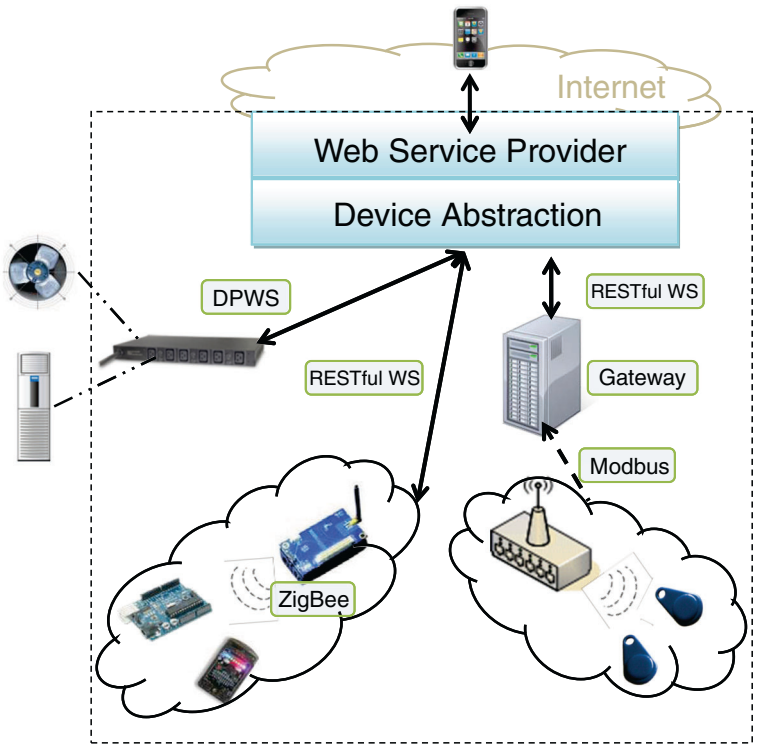


Figure 6. The remote appliance management prototype.

Security Comm. Networks 2014; **7**:2696–2712 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/sec

6.4. Policy specification

Label policies. We can assign permission labels to each of the objects, $o \in$ {serviceOn1,serviceOn2,-serviceOff1,serviceOff2}, with Function 11: $\varpi_{o_1}(o) :=$ *cn.pku.sei.o.INVOKE*. In addition, we can also assign the permission labels to the subject *ApplicationGUID* : ϖ_s(*ApplicationGUID*) : = *P*.

Usage control policy. According to Section 5.4, the administrator can specify his polices with XML, as illustrated in Figure 7. The root node < Policies > contains all the policies. It includes several < Policy > tags; each indicates a user-specified policy defined in Definition 15.

6.5. Usage decision

Authorization. Once the application invokes the web service *ServiceOn1*, the authorization Ω(*ApplicationGUID*, *ServiceOn*1, *invoke*) (see Definition 16) will be performed.

Usage decision. A usage decision contains three stages, as defined in Definition 17:

1. Before the usage, the pre-obligation constraint, precontext constraint, and state constraint specified in Figure 7 are evaluated. If all of them are satisfied, *allow*(*ApplicationGUID*, *ServiceOn*1, *invoke*) is invoked;
2. After Step 6.5.0.10, if the usage is allowed, $update((ServiceOn1, startTime, \upsilon(ServiceOn1, startTime))$ will be called, that is, $\mu(ServiceOn1, startTime, \upsilon(ServiceOn1, startTime)) = system.currentTime$;
3. After Step 6.5.0.10, if the usage is allowed, the ongoing constraints are evaluated continuously. If any of ongoing policies is violated, *revoke*(*ApplicationGUID*, *ServiceOn*1, *invoke*) will be called.

In summary, we have built a context-aware usage control framework prototype for the remote appliance management system, to enable family members to remotely manipulate household appliances in a context-aware manner. This case study demonstrates that we can utilize ConUCON to enhance privacy and security protection in a real WoT system.

7. RELATED WORK

There have been many research efforts in building infrastructures for IoT, as well as for WoT, RFID, and WSN. Roman *et al.* [28] have surveyed existing work in securing IoT infrastructure and discussed future challenges that will be encountered in this area. RFID security [10] has become a hot topic recently, with the aim to protect RFID security and privacy against the cloning attack [52,53], physical attacks [54,55], relay attack [56], electronic collisions [57], etc. For WSN, The following security aspects have been studied [58]: routing [59], data aggregation, trust

```
<Policies>
    <Policy Effect="Permit">
        <Subject>All</Subject>
        <Object>ServiceOn1</Object>
        <Right>Invoke</Right>
        <Obligations>
            <Obligation ObligationTime="Previous" ObligationID = "cn.pku.sei.obligation1"></
                Obligation>
            <Obligation ObligationTime="Ongoing" ObligationID = "cn.pku.sei.obligation2"></
                Obligation>
        </Obligations>
        <States>
            <State>
                <Attribute Owner="ServiceOn1" Type = "startTime"></Attribute>
                <Expression>System.currentTime - startTime >= 30</Expression>
            </State>
        </States>
        <Contexts>
            <Context ContextTime="Previous">
                <ContextComposition Operator="~">
                    <Factor Type="Temporal">[2011-01-01T00:00:00,2011-12-31T23:59:59],weeks
                        +{1,2,...,5}.day+9 hours->8hours</Factor>
                    <Factor Type="Spatial">vehicle</Factor>
                    <Factor Type="Spatial">office</Factor>
                </ContextComposition>
            </Context>
            <Context ContextTime="Ongoing">
                <Factor Type="temperature">temperature>=26</Factor></Context>
        </Contexts>
        <Updates>
            <Update UpdateTime="Start">
                <Attribute Owner="ServiceOn1" Name = "StartTime"></Attribute>
                <Expression>ServiceOn1 := System.currentTime</Expression>
            </Update>
        </Updates>
    </Policy>
</Policies>
```

Figure 7. An XML representation of the policies.

management [13,14], etc. In both RFID and WSN, encryption plays an important role because it serves as the foundation of many security schemes such as authentication, signature, etc. Unfortunately, traditional encryption algorithms consume a great amount of resources, making them inappropriate for smart objects such as sensors. In response to this issue, several lightweight solutions have been proposed [60,20–22].

Access control models play an important role in security mechanisms. Some researchers have extended the RBAC model [61] to include context information in authorization decisions. Damiani *et al*. proposed GEO-RBAC [39] to support spatial roles. Bertino *et al*. proposed TRBAC [41] to support temporal roles. Other extensions include GRBAC [62,63], STARBAC [64], and LRBAC [65]. To the best of our knowledge, only a couple of solutions have been proposed to perform access control for IoT. Ngo *et al*. [27] proposed an authorization control model PRBAC based on RBAC to enhance the authorization control for the wireless network services. By combining the WoT with social networks, Guinard *et al*. [66] adopt authentication mechanisms existing in social networks to perform access control in the WoT.

The UCON model [31–34] is a generic security model that meets a wide range of security requirements. Introducing UCON model into the WoT can provide several benefits. On one hand, it is able to describe nearly all the permission models, such as the permission label model in Android and the Information flow model, which is more appropriate for WoT than the RBAC models. On the other hand, it supports continuous evaluation (i.e. ongoing usage decision in UCON), which inherently copes with the dynamic nature of contexts. As an extension of UCON, ConUCON not only inherits the UCON's advantages but also enhances the context definition and context-aware policy specification. CA-RBAC [16] is another work that provides continuous evaluation. However, it only supports role-based access control, which limits its users in terms of permission model adoption.

8. CONCLUSION

To provide a seamless integration with the Web, the WoT integrates the heterogeneous underlying systems in the current IoT, and exposes the resulting web services on the Web. Because of several inherent characteristics, WoT requires additional security and privacy protection beyond the existing protection in traditional computers.

This paper proposes ConUCON, which leverages the context information to enhance data, resource, and service protection in the WoT. On the basis of the ConUCON, we also design and implement a context-aware usage control framework in the middleware level of our ongoing SmartHome project. Finally, we utilize the ConUCON in a remote appliance management prototype, as a case study, to demonstrate its applicability and feasibility in a real WoT applicaiton.

ACKNOWLEDGEMENTS

We are grateful to the guest editor and anonymous reviewers for their valuable comments, suggestions, and effort on this paper. This work is supported partly by the National High Technology Research and Development (863) Program of China under Grant No. 2011AA01A202, the National Basic Research Program (973) of China under Grant No. 2009CB320703 the Science Fund for Creative Research Groups of China under Grant No. 60821003, the National Natural Science Foundation of China under Grant No.61103026, and a PKU-IBM joint project.

REFERENCES

1. International Telecommunication Union. Itu internet report 2005: The internet of things. 2005.
2. Guinard D, Trifa V. Towards the web of things: Web mashups for embedded devices. In *Workshop MEM 2009, in proceedings of WWW*, Madrid, Spain, 2009.
3. Dawson-Haggerty S, Jiang X, Tolle G, Ortiz J, Culler D. smap: a simple measurement and actuation profile for physical information. In *Proceedings of SenSys '10*, New York, USA, ACM, 2010; 197–210.
4. Sá De Souza LM, Spiess P, Guinard D, Köhler M, Karnouskos S, Savio D. Socrades: a web service based shop floor integration infrastructure. In *Proceedings of IOT'08*, Springer-Verlag, Berlin, Heidelberg, 2008; 50–67.
5. Information Society Technologies. Working group report on web-based service industry. 2008.
6. Rieback MR, Simpson PN, Crispo B, Tanenbaum AS. RFID malware: design principles and examples. *Pervasive and Mobile Computing* 2006; **2**(4):405–426.
7. Rieback MR, Crispo B, Tanenbaum AS. Is your cat infected with a computer virus? In *Proceedings of PerCom'06*, Washington, DC, USA, 2006; 169–179.
8. Brownfield M, Gupta Y, Davis N. Wireless sensor network denial of sleep attack. 2005; 356–364.
9. Rekleitis E, Rizomiliotis P, Gritzalis S. An agent based back-end RFID tag management system. In *Proceedings of TrustBus'10*, Springer-Verlag: Berlin, Heidelberg, 2010; 165–176.
10. Juels A. RFID security and privacy: a research survey. *IEEE Journal on Selected Areas in Communications* 2006; **24**(2):381–394.
11. Chen X, Makki K, Yen K, Pissinou N. Sensor network security: a survey. *Communications Surveys & Tutorials, IEEE* 2009; **11**(2):52–73.
12. Perrig A, Stankovic J, Wagner D. Security in wireless sensor networks. *Communications of the ACM* 2004; **47**:53–57.
13. Lopez J, Roman R, Agudo I, Fernandez-Gago C. Trust management systems for wireless sensor networks: best practices. *Computer Communications* 2010; **33**:1086–1093.

14. Yang Y, Zhou J, Deng RH, Bao F. Better security enforcement in trusted computing enabled heterogeneous wireless sensor networks. 2010.
15. Al-Muhtadi J, Ranganathan A, Campbell R, Mickunas MD. In Proceedings of the First IEEE International Conference on Pervasive Computing and Communications. 2003; 489–496.
16. Kulkarni D, Tripathi A. Context-aware role-based access control in pervasive computing systems. In *Proceedings of the 13th ACM symposium on Access control models and technologies*, SACMAT '08, New York, NY, USA, ACM, 2008; 113–122.
17. Corradi A, Montanari R, Tibaldi D. Context-based access control for ubiquitous service provisioning. In *Proceedings of the 28th Annual International Computer Software and Applications Conference*, COMPSAC '04, Washington, DC, USA, 2004; 444–451.
18. Andreini F, Crisciani F, Cicconetti C, Mambrini R. Context-aware location in the internet of things. In *GLOBECOM Workshops*. IEEE: Miami, FL, 2010; 300–304.
19. Simoes J, Magedanz T. Contextualized User-Centric Multimedia Delivery System for Next Generation Networks. *Telecommunication Systems*, 2010; 1–16.
20. Roman R, Alcaraz C. Applicability of public key infrastructures in wireless sensor networks. In *EuroPKI'07*, 2007; 313–320.
21. Roman R, Alcaraz C, Lopez J. A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes. *Mobile Networks and Applications*, IEEE: Miami, FL, 2007; **12**:231–244.
22. Chu C-K, Liu JK, Zhou J, Bao F, Deng RH. Practical id-based encryption for wireless sensor network. In *Computer and Communications Security*. ACM: N.Y. USA, 2010; 337–340.
23. He W, Liu X, Nguyen H, Nahrstedt K, Abdelzaher TT. Pda: Privacy-preserving data aggregation in wireless sensor networks. In *INFOCOM 2007, IEEE*, 2007; 2045–2053.
24. Carbunar B, Yu Y, Shi W, Pearce M, Vasudevan V. Query privacy in wireless sensor networks. *ACM Trans. Sen. Netw.* 2010; **6**:14:1–14:34.
25. Sen J. Privacy preservation technologies in internet of things. *CoRR*, 2010; 1–1.
26. Welbourne E, Battle L, Cole G, *et al.* Building the Internet of Things using RFID: the RFID ecosystem experience. *Internet Computing, IEEE* 2009; **13**(3):48–55.
27. Ngo HH, Wu XP, Le PD, Wilson C. Package-role based authorization control model for wireless network services. In *ARES '09*, 2009; 475–480.
28. Roman R, Najera P, Lopez J. Securing the internet of things. *IEEE Computer* 2011; **44**(9):51–58.
29. Fielding RT. Architectural styles and the design of network-based software architectures. PhD thesis, 2000. AAI9980887.
30. Priyantha NB, Kansal A, Goraczko M, Zhao F. Tiny web services: design and implementation of interoperable and evolvable sensor networks. In *Proceedings of the ACM SenSys '08*, New York, USA, ACM, 2008; 253–266.
31. Park J, Sandhu RS. Towards usage control models: beyond traditional access control. In *SACMAT*, 2002; 57–64.
32. Sandhu R, Park J. Usage control: A vision for next generation access control. In *MMMACNS'03, LNCS*, 2003.
33. Park J, Sandhu R. The UCON_{ABC} usage control model. *ACM Transactions on Information and System Security* 2004; **7**(1):128–174.
34. Zhang X, Parisi-Presicce F, Sandhu R, Park J. Formal model and policy specification of usage control. *ACM TISSEC* 2005; **8**(4):351–387.
35. Bai G, Feng T, Gu L, Guo Y, Chen X. Context-aware usage control for android. In *Security and Privacy in Communication Networks, volume 50 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer: Berlin, Heidelberg, 2010; 326–343.
36. Myers AC, Liskov B. Complete, safe information flow with decentralized labels. In *S&P 1998, IEEE*, 1998; 186–197.
37. Android reference. Security and Permissions. http://developer.android.com/guide/topics/security/security.html.
38. Chen G, Kotz D. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College, Computer Science, Hanover, NH, November 2000.
39. Damiani ML, Bertino E, Catania B, Perlasca P. Geo-rbac: a spatially aware rbac. *ACM Transactions on Information and System Security* 2007; **10**(1):2.
40. ISO 8601. Data elements and interchange formats—information interchange—Representation of dates and times. 2004; http://www.iso.org/iso/catalogue_detail?csnumber=40874, August 10, 2011.
41. Bertino E, Bonatti PA, Ferrari E. TRBAC: a temporal role-based access control modelIn. In *Proceedings of RBAC-00*. ACM Press: N.Y., 2000; 21–30.
42. Bonato P. Advances in wearable technology and applications in physical medicine and rehabilitation. *Journal of Neuroengineering and Rehabilitation* 2005; **2**(1):2.
43. Ptaszynski M, Dybala P, Shi W, Rzepka R, Araki K. Towards context aware emotional intelligence in machines: computing contextual appropriateness of affective states. In *Proceedings of the IJCAI'09*, San Francisco, CA, USA, 2009; 1469–1474.
44. Lee JKH. Combining context-awareness with wearable computing for emotion-based contents service. *International Journal of Advanced Science and Technology* 2010; **22**, http://www.sersc.org/journals/IJAST/vol22/2.pdf.

45. Bell D, LaPadula L. Secure computer systems: Mathematical foundations. Technical Report ESD-TR-73-278, MITRE Corporation, 1973.
46. Biba KJ. Integrity considerations for secure computer systems. MTR-3153, Rev. 1, The Mitre Corporation, 1977.
47. XACML (eXtensible Access Control Markup Language). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml, March 25, 2011.
48. SAML (Securuty Assertion Markup Language). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security, March 25, 2011.
49. Web Services Policy 1.5—Framework. 2007; http://www.w3.org/TR/2007/REC-ws-policy-20070904/, March 25, 2011.
50. Anagnostopoulos CB, Ntarladimas Y, Hadjiefthymiades S. Situational computing: an innovative architecture with imprecise reasoning. *Journal of Systems and Software* 2007; **80**:1993–2014.
51. Brunato M, Battiti R. Statistical learning theory for location fingerprinting in wireless lans. *Comput. Netw. ISDN Syst.* 2005; **47**:825–845.
52. Tuyls P, Batina L. RFID-Tags for Anti-counterfeiting. In *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *LNCS*, 2006; 115–131.
53. Skorobogatov SP, Anderson RJ. Optical fault induction attacks. Springer-Verlag, 2002; 2–12.
54. Juels A, Rivest RL, Szydlo M. The blocker tag: selective blocking of RFID tags for consumer privacy. In *Proceedings of the CCS '03*, New York, NY, USA, ACM, 2003; 103–111.
55. Weis SA, Sarma SE, Rivest RL, Engels DW. Security and privacy aspects of low-cost radio frequency identification systems. 2003. http://www.sersc.org/journals/IJAST/vol22/2.pdf
56. Fishin K, Roy S, Jiang B, Fishkin KP, Roy S, Jiang B. Some Methods for Privacy in RFID Communication. 2004.
57. Wu VKY, Campbell RH. Using generalized query tree to cope with the capture effect in RFID singulation. In *Proceedings of CCNC'09*, Piscataway, NJ, USA. IEEE, 2009; 516–520.
58. Walters JP, Liang Z, Shi W, Chaudhary V. Wireless sensor network security: A survey. In *in Distributed, Grid, and Pervasive Computing, Yang Xiao (Eds)*, CRC Press, 2007; 0–849.
59. Deng J, Han R, Mishra S. Insens: Intrusion-tolerant routing in wireless sensor networks. 2002.
60. Staake T, Thiesse F, Fleisch E. Extending the EPC network: the potential of RFID in anti-counterfeiting. In *Proceedings of SAC '05*, New York, NY, USA, ACM, 2005; 1607–1612.
61. Sandhu RS. Role-based access control. *Advances in Computers* 1998; **46**:238–287.
62. Moyer MJ, Abamad M. Generalized role-based access control. In *Distributed Computing Systems, 2001*. Mesa, AZ , USA 2001; 1391–398.
63. Covington MJ, Moyer MJ, Ahamad M. Generalized role-based access control for securing future applications, 2000.
64. Aich S, Sural S, Majumdar AK. STARBAC: spatio temporal role based access control. In *OTM Conferences (2)*, volume **4804**, Springer, 2007; 1567–1582.
65. Ray I, Kumar M, Yu L. LRBAC: A location-aware role-based access control model. In *ICISS*, volume 4332 of *Lecture Notes in Computer Science*, Springer: 2006; 147–161.
66. Guinard D, Fischer M, Trifa V. Sharing using social networks in a composable web of things. In *IEEE ICPCC*, 2010; 702–707.