An Axiomatic Category-Based Access Control Model for Smart Homes

Clara Bertolissi[1][0000−0001−9283−1386], Maribel Fernández[2][0000−0002−1959−8730], and Bhavani Thuraisingham[3]

[1] Aix-Marseille Université, CNRS LIS UMR 7020, France
Clara.Bertolissi@univ-amu.fr
[2] King's College London, United Kingdom,
maribel.fernandez@kcl.ac.uk
[3] University of Texas at Dallas, USA
bxt043000@utdallas.edu

Abstract. Internet of Things (IoT) refers to a system of devices that send and receive data via the internet. In the Smart Home IoT, appropriate access controls are key for the security of the household. Popular access control models, such as RBAC and ABAC, have been adapted to this context, but user studies show that new hybrid models are required. We propose a logic-based access control model that is highly expressive: it subsumes the RBAC and ABAC models as well as a whole spectrum of hybrid models. Policies are specified via categorisation of users and devices (a natural mechanism for smart home owners) and have a logic semantics that facilitates policy verification. We have identified a simple yet expressive submodel that satisfies the criteria highlighted in user studies for smart home access control.

Keywords: Access Control · Internet of Things · Smart Home · Category-Based Access Control

1 Introduction

The Smart Home Internet of Things (IoT) consists of a network of devices (home appliances equipped with sensors and software, such as smart locks, TV, playstation, etc.) that can be controlled remotely via web and mobile applications. Access to devices is usually granted to household members (which may include children, babysitters, visiting relatives, etc.) under certain conditions, specified via access control policies. Popular access control models, such as Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC), have been adapted to the Smart Home IoT scenario [2, 3], however, user studies [17] show that the distinctive features of smart homes require new models combining the benefits of RBAC and ABAC. Indeed, RBAC policies, where users rights are defined on the basis of their role in the household (e.g., parent, child, babysitter) are easy to specify and review, but are not sufficiently flexible to take into account dynamic features (such as whether the babysitter is actually at home

or not). ABAC policies assign users rights based on attribute values of users, devices and the environment, and permissions can automatically change when attribute value changes, making it more suitable to the dynamic smart home environment. However, ABAC policies are notoriously difficult to manage. Recent research [3, 4] suggests that a hybrid model combining the benefits of RBAC and ABAC is required for smart homes.

Two hybrid models, one based on RBAC and the other on ABAC, have been proposed to address this need: $HyBAC_{RC}$ and $HyBAC_{AC}$ [4]. However, as stated by the authors [4], the "ultimate goal is to have a family of access control models ranging from relatively simple to more sophisticated [...] to provide policy designers with a range of models to choose from according to the environment requirements and the business needs".

In this paper, we propose to achieve this goal via Category-Based Access Control (CBAC). We define a general model, called *SHoCBAC*, from which a family of instances can be obtained ranging from simple RBAC-style models to sophisticated ABAC-style ones and covering the whole spectrum in between. An advantage of this unified approach, where all the models are obtained as instances of *SHoCBAC* by defining appropriate notions of categories, is that policy languages, enforcement mechanisms and policy analysis techniques can be defined once for *SHoCBAC* and then shared across all the instances, thus saving effort. Moreover, cognitive science indicates that categorisation is one of the main mechanisms to organise knowledge [13, 6] and user studies confirm that category-based specifications are easier to manage than rule-based ones [19], making *SHoCBAC* well-suited for smart homes.

SHoCBAC is an axiomatic model with a formal semantics that makes it possible to reason about policies. As all CBAC models, it can be equipped with a rewriting-based operational semantics, which facilitates the analysis of policies (e.g., to verify consistency) [10, 1].

Since administrative policies can also be defined in an instance of the CBAC metamodel, we are also able to define administrative policies for the Smart Home IoT in the same framework.

With the aim of simplifying policy creation and maintenance tasks for smart home owners, we have identified a core policy template with one administrative category and three basic user categories based on notions of trust, which can be parameterised to take into account environmental attributes. Actions for devices are categorised as safe or unsafe (e.g., opening a door lock or changing the oven temperature are unsafe actions, whereas turning off a light is safe) and assigned to user categories. We show that the core template covers most of the access control requirements for smart homes [17], while the general model, *SHoCBAC*, is able to express all the policies that can be expressed in previous models for Smart Home IoT [4].

The paper is organised as follows: Section 2 provides preliminary notions on category-based access control, Section 3 introduces *SHoCBAC* and Section 4 shows its expressive power by providing encodings of existing models. Section 5 discusses related work and Section 6 concludes.

2 Preliminaries: The CBAC Metamodel

In this section we recall the main concepts underlying the category-based access control metamodel [7]. We assume familiarity with first-order logic.

The CBAC metamodel aims at facilitating the definition of access control models. It consist of a family $\mathcal{E}$ of sets of **entities**, which are classified into **categories**, a family $\mathcal{R}el$ of **relationships between entities**, and a set $\mathcal{A}x$ of **axioms** that specify the properties that the model must satisfy.

The classification of entities into categories can be static (e.g., categories can be defined in terms of roles, which can only be updated by the administrator) or can be defined in terms of dynamic parameters (e.g., an airline may categorise clients in terms of the miles travelled, which can be automatically updated each time a user validates new air miles). Since categories can be application dependent, CBAC is extremely expressive: most of the existing access control models can be defined as instances of CBAC by selecting appropriate sets of entities and relationships and specifying adequate notions of categories (see [7, 9]).

The metamodel includes the following generic sets of entities in addition to application-dependent entities: a countable set $\mathcal{C}$ of categories, denoted c_0, c_1, ...; a countable set $\mathcal{P}$ of principals, denoted p_0, p_1, ... (we assume that principals that request access to resources are pre-authenticated); a countable set $\mathcal{A}$ of named *actions*, denoted a_0, a_1, ...; a countable set $\mathcal{R}$ of *resource identifiers*, denoted r_0, r_1, ...; a finite set $\mathcal{A}uth$ of possible *answers* to access requests (e.g., {grant, deny, undetermined}).

The metamodel includes the following generic relationships:

- *Principal-Category Assignment*, $\mathcal{PCA} \subseteq \mathcal{P} \times \mathcal{C}$, which assigns categories to principals: $(p, c) \in \mathcal{PCA}$ iff the principal $p \in \mathcal{P}$ is in the category $c \in \mathcal{C}$;
- *Resource-Category Assignment*, $\mathcal{RCA} \subseteq \mathcal{R} \times \mathcal{C}$, which assigns categories to resources: $(r, c) \in \mathcal{RCA}$ iff the resource $r \in \mathcal{R}$ is in the category $c \in \mathcal{C}$;
- *Permissions*, $\mathcal{ARCA} \subseteq \mathcal{A} \times \mathcal{C} \times \mathcal{C}$, which assigns permitted actions to categories of principals and resources; such that $(a, c_r, c_p) \in \mathcal{ARCA}$ iff the action $a \in \mathcal{A}$ on resource category $c_r \in \mathcal{C}$ can be performed by principals assigned to the category $c_p \in \mathcal{C}$.
  Similarly, banned actions are specifed via the relation $\mathcal{BARCA}$.
- *Authorisations*, $\mathcal{PAR} \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{R}$, such that $(p, a, r) \in \mathcal{PAR}$ iff the principal $p \in \mathcal{P}$ is allowed to perform the action $a \in \mathcal{A}$ on the resource $r \in \mathcal{R}$.
  Similarly, a relation $\mathcal{BAR}$ defining prohibitions is also included.

Additional application-dependent relations can also be included. In this paper, we consider also categories of actions (safe/unsafe), and a relation $\mathcal{ACA} \subseteq \mathcal{A} \times \mathcal{C}$ (action-category assignment). In addition, the metamodel includes a reflexive-transitive relation $\subseteq$ between categories to specify a hierarchy of categories and permission inheritance.

Authorisations are directly deduced from the previous relations using the **core axiom** defined below (more axioms are added to specify how prohibitions are deduced from $\mathcal{BARCA}$ and to ensure that no inconsistencies arise; we refer to [10] for details).

$(a1)\ \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R}, (\exists c_p \in \mathcal{C}, \exists c'_p \in \mathcal{C}, \exists c_r \in \mathcal{C}, \exists c'_r \in \mathcal{C}, \exists c_a \in \mathcal{C},$
$(p, c_p) \in \mathcal{PCA} \wedge (r, c_r) \in \mathcal{RCA} \wedge (a, c_a) \in \mathcal{ACA}$
$\wedge\ c_p \subseteq c'_p \wedge c_r \subseteq c'_r \wedge (c_a, c'_r, c'_p) \in \mathcal{ARCA}) \Leftrightarrow (p, a, r) \in \mathcal{PAR}$

Based on this axiom, we can deduce that if a principal p is in the category c_p (that is, $(p, c_p) \in \mathcal{PCA}$), a resource r is in the category c_r (that is, $(r, c_r) \in \mathcal{RCA}$), and the category c_p is permitted to perform actions from category c_a on resource category c_r (that is, $(c_a, c_r, c_p) \in \mathcal{ARCA}$) then p is authorised to perform a on r (that is, $(p, a, r) \in \mathcal{PAR}$).

Definition 1. *Given a specification of $\mathcal{E}$ and $\mathcal{R}el$, and a set $\mathcal{A}x$ of axioms, a CBAC* ***policy*** *is a tuple $\langle E, Rel\rangle$ that defines the contents of $\mathcal{E}$ and $\mathcal{R}el$ such that E, Rel satisfy all the axioms in $\mathcal{A}x$.*

Sessions are not included in the core CBAC model but if needed they could be added as another set of entities, or they could be specified via user attributes.

RBAC and ABAC as instances of CBAC. If categories of principals are defined using roles and resources are not categorised (i.e., each resource is in its own individual category) then we obtain the standard RBAC model. ABAC policies can be obtained by using a notion of category based on user, resource and environment attributes. C-ABAC [14] is a formal specification of the ABAC model in the CBAC metamodel. In C-ABAC, the set $\mathcal{E}$ of entities includes, in addition to the generic sets $\mathcal{P}$, $\mathcal{A}$, $\mathcal{R}$, $\mathcal{C}$, a countable set $\mathcal{E}nv$ of environment entities (e.g., networks, clock, etc.) and a set $\mathcal{A}t$ of *attributes*, ranging over *values* $\mathcal{V}$. A set $\mathcal{C}ond$ of Boolean expressions involving attributes and values is used to define categories: the expressions specify the acceptable ranges of attribute values required by each category. The set $\mathcal{R}el$ of relationships in C-ABAC includes:

- *Principal-Attribute Assignment*, $\mathcal{PAtA} \subseteq \mathcal{P} \times (\mathcal{A}t \times \mathcal{V})^*$, which defines the attributes of each principal and their current values;
- *Resource-Attribute Assignment*, $\mathcal{RAtA} \subseteq \mathcal{R} \times (\mathcal{A}t \times \mathcal{V})^*$, which defines the attributes of each resource and their current values;
- *Environment-Attribute Assignment*, $\mathcal{EAtA} \subseteq \mathcal{E}nv \times (\mathcal{A}t \times \mathcal{V})^*$, which defines the relevant environmental attributes and their values;
- *Category-Attribute Assignment*, $\mathcal{CAtA} \subseteq \mathcal{C} \times \mathcal{C}ond$, which defines a condition on attribute values for each category. We write $\Gamma \vdash cond$ if Γ specifies attributes and values that satisfy *cond*. C-ABAC includes, in addition to the core axiom $(a1)$, also categorisation axioms (see [14] for more details).

$(c1)\ \forall p \in \mathcal{P},\ \forall c \in \mathcal{C},\ ((p, c) \in \mathcal{PCA} \Leftrightarrow$
$\exists l_p, (p, l_p) \in \mathcal{PAtA}, \exists cond, (c, cond) \in \mathcal{CAtA}, (l_p, \mathcal{RAtA}, \mathcal{EAtA} \vdash cond))$
$(c2)\ \forall r \in \mathcal{R},\ \forall c \in \mathcal{C},\ ((r, c) \in \mathcal{RCA} \Leftrightarrow$
$\exists l_r, (r, l_r) \in \mathcal{RAtA}, \exists cond, (c, cond) \in \mathcal{CAtA}, (l_r, \mathcal{PAtA}, \mathcal{EAtA} \vdash cond))$

An administrative model for access control, *Admin-CBAC*, has also been defined as an instance of the CBAC metamodel (see [11]). For this, the sets of entities and relationships are split into standard and administrative subsets, and administration axioms are included to ensure safety.

3 Smart Home Instance of CBAC: *SHoCBAC*

Authorisation and privacy issues in the smart home domain have attracted attention recently. The threat model includes two major kinds of adversaries: external third parties that try to access devices or the data produced by the devices and insiders that misuse the devices or their generated data, with the intent of causing physical, financial or privacy-related damage.

To avoid device misuse, access control policies for the smart home need to be fine grained (to specify rights at the level of device actions rather than the device as a whole) and need to be dynamic to take into account environmental conditions, such as time and location. CBAC permits the definition of fine-grained, dynamic authorisation at the level of actions, has a formal definition and several proof of concept implementations exist (see [11]). Additionally, its category-based foundations make it easy to combine it with data sharing models [15], to offer not only access control but also privacy guarantees as required. These features make CBAC a suitable approach for the smart home domain.

In this section we define an instance of CBAC, *SHoCBAC*, for the Smart Home IoT domain. Categories will be specified using attributes, so *SHoCBAC* is actually an instance of C-ABAC, as specified in Section 2. We start by specifying the sets of entities and relationships considered.

SHoCBAC Entities: The set $\mathcal{P}$ of principals in *SHoCBAC* includes smart home users as well as services that require access to data generated by smart home devices. The set $\mathcal{R}$ of resources include devices and their generated data. The set $\mathcal{A}$ of actions includes actions on devices (such as play for entertainment devices, on/off for lights and locks, etc.) as well as actions that services can request on data, such as read (electricity consumption data or lock log status, for example). The set of attributes $\mathcal{A}t$ includes *role* for principal (e.g., owner, child, babysitter), and *type* for resources (e.g., entertainment device, security device). We consider environmental attributes in $\mathcal{EAtA}$: *date*, *time*, and additionally *weekDay*, *weekEnd*, *nightTime* and *dayTime* that can be customised by the owner to denote dates and times in certain ranges. For example, *nightTime* can be defined to be $\{21.00 - 7.00\}$, and *weekDay* to be *{Monday, Tuesday, Wednesday, Thursday, Friday}*. Similarly, we use the predefined Boolean attribute *holiday*, which is also customisable and it has the value true when the owners are on holidays. We also consider an additional attribute *onwer@Home*, which is a Boolean and has the value true if one of the owners is at home.

Policies in *SHoCBAC* include predefined sets of categories:

- Principal Categories: There are two core categories of principals: *Users*, which may have subcategories such as home-owner, partner, child, babysitter, neighbour, visitor, and *Services* which may have subcategories such as health, insurance, energy-company, etc.
- Resource Categories: There are two core categories of resources: *Data*, which may have subcategories such as log-files, usage time, etc., and *Devices*, which has subcategories such as entertainment, cooking, etc.

– Action Categories: There are two core categories of actions: *Safe* and *Unsafe*. Even though in CBAC actions are not usually categorised, in the smart home domain it is convenient to distinguish categories of actions to facilitate the definition of policies (see examples below).

Although we assume two core categories for principals and two for resources (following a separation of concerns principle), it is possible for a household to define only a policy to control access to devices without any restriction on the use of data and vice-versa only a policy to restrict access to data but no restriction on the use of devices. In the rest of the paper we focus on the access control sub-model, that is we will not consider the subcategories Data and Services.

SHoCBAC Relationships and Axioms: The sets of relationships and axioms in *SHoCBAC* are the ones defined for C-ABAC.

SHoCBAC may be seen also as an instance of *Admin-CBAC*, including an administrative level where there is only one category: *owner-admin*. If administrative policies are required, the additional relationships and axioms in *Admin-CBAC* should be included.

We do not consider sessions in this paper, because they are not considered an essential feature of Smart Home IoT systems in user studies (see e.g. [17]). If needed, sessions can be modelled as an attribute of users in C-ABAC.

3.1 Use Case

We illustrate the use of *SHoCBAC* with an example, inspired from the user study in [17]. In this work, participants give their access control preferences for a list of pre-identified capabilities, according to the following characteristics: teenager, child, visiting family, babysitter, neighbour. We describe below some examples and show how they can be modelled using *SHoCBAC*.

Example 1 (Static Policy). Consider a family composed of Omar, who is the owner of the smart devices and administrator of the smart hub used to access the devices, and partner Oprah, who hired a babysitter (Bea) to take care of two children, Clohe aged 8 and Tom aged 14. They have several neighbours, of which Neil is trusted to take care of the house when the family is on holidays. They also host some other members of the family (e.g. aunt Vicky) who come to visit from time to time.

This can be specified in *SHoCBAC* by including in the set of principals all the members of the family, assigning them a role (e.g., owner, child, ...) and defining a set of (static) categories based on their role $\mathcal{C}$ ={owner, teenager, child, supervisedChild, houseKeeper, visitor, babysitter, neighbour}. Thus, we obtain the following $\mathcal{PCA}$ relation:

$\mathcal{PCA}$ ={(Omar, owner), (Oprah, owner), (Clohe, child), (Bea, Babysitter), (Tom, teenager), (Neil, neighbour), (Vicky, visitor)}.

We can suppose for now that principal categories are statically assigned by Omar, acting as the administrator of the policy.

The house is equipped with several IoT connected devices, such as a voice assistant, a smart TV, a playstation, smart heating, lights and cameras, smart oven, etc. These devices form the set $\mathcal{R}$ of resources of the *SHoCBAC* policy. Each device has a type attribute and a standard set of actions: open/close, lock/unlock, turn on/off, play, etc., which are categorised as safe or unsafe (e.g., turning the oven on, changing its temperature are considered unsafe, whereas turning the oven off is safe). Different device categories are defined on the basis of the device type: entertainment, cooking, lights, heating, etc.

Again, we can assume that device categories are static, based on the device type, which is assigned by the administrator. Thus, the $\mathcal{RCA}$ relation includes for instance: $\mathcal{RCA}$ ={ (oven, Cooking), (TV, Entertainment), ... }. The administrator of the policy (i.e. Omar) can then define permissions over categories of users and devices (the $\mathcal{ARCA}$ relation of the *SHoCBAC* policy). The use of categories of actions simplifies the definition of permissions, since it is possible for example to assign to the Child category the permission to perform any safe action on the cooking devices: $\mathcal{ARCA}$ = { (SafeCooking, Cooking, Child), (UnsafeHeat, Heating, Owner),... }.

To take into account the fact that authorisations may change depending on factors such as whether the family is on holidays or not, we can use dynamic attributes, such as time and location, in the definition of categories. Attributes are seen as functions returning a value at query evaluation time. In the following, we use the notation *entity.attribute* to denote principal, resource and environment attributes. Recall that in C-ABAC, category definitions consist of Boolean expressions. We give an example next.

Example 2 (Dynamic Policy). Categories as defined in Example 1 can be refined by using other attributes in addition to *role* for principals and *type* for resources, in such a way that authorisations adapt to changes in the environment without the need of manual changes by the administrator. For instance, we can refine the static category Child defined in Example 1, by defining a sub-category *SupervisedChild* that includes a condition about the presence of one of the parents at home, which may imply more permissions for the child. For this, we use the environment attribute *owner@Home*, which is true if one of the owners is at home (i.e., at a GPS location corresponding to the house).

$$SupervisedChild.cond = (p.role = Child \text{ and } env.owner@Home)$$

In this way, the $\mathcal{PCA}$ relation will change dynamically (unlike the $\mathcal{PCA}$ relation specified in Example 1, which is static):

$$\mathcal{PCA}(p) = \text{if } p.role = Child \text{ then (if } env.owner@Home \text{ then } SupervisedChild \text{ else } p.role)$$

Similarly, we may want to define permissions according to specific intervals of time. To this end, we can use the environmental attributes defined above to specify, e.g. that children can turn on the TV only on weekends, or that

neighbours can have access to the house when the owner is on holidays. For example:

$$\mathcal{PCA}(p) = \mathsf{if}\ p.role = Neighbour\ \mathsf{then}\ (\mathsf{if}\ env.holiday\ \mathsf{then}\ HouseKeeper\ \mathsf{else}\ ...$$

where the category *HouseKeeper* has additional permissions w.r.t. the category *Neighbour*, such as access to Security devices (alarm, cameras, door lock, etc).

We choose to specify restrictions using conditions in the definition of the category, rather than adding an ad-hoc condition in the definition of the permissions (i.e. $\mathcal{ARCA}$ in our case, or role-permission assignment in a RBAC style model [2]). In this way, we can obtain a structured policy where principals (resp. resources) that share the same permissions are grouped in a category.

Remark: Administrative categories could be similarly defined. For example, we could have several administrators (Ad1, Ad2,...) associated with different administrative categories (Entertainment-mngmnt, Ownership-control,...). We could also include prohibitions (e.g., to forbid administrators from making certain permission assignments) by using the relation $\mathcal{BARCA}$. In this way we could for example specify that the administrator Ad1 cannot add a permission for the Child category to access the Entertainment device category on school days.

3.2 Core *SHoCBAC* Policy Template

The above example highlights the expressive power of *SHoCBAC*. In this section we focus on another important property of access control models: usability. User studies show that access control models should align with natural cognitive processes [19]. Moreover, in the case of smart homes, simplicity is key [17]. We will define a simple *SHoCBAC* policy template, *Core*, that has both these features and is sufficiently expressive for Smart Home applications.

Minimal template. Cognitive science studies [13, 6] confirm that categorisation is one of the main mechanisms underlying cognition, and categorisation is easier if there is only a small number of groups. Taking this into account, we propose a default policy template with only one administrative category (owner) and three standard principal categories defined on the basis of trust levels. More precisely, each of the three principal categories, LT, MT, FT, are associated with a trust level, namely *low, medium* and *full* trust. Trust level is an attribute of principals (i.e., it will be recorded in $\mathcal{PAtA}$). Typically, the owners have full trust level, external people coming regularly into the house (e.g., for working reasons, such as babysitters and house cleaners, or to visit, such as grand parents), have medium trust level, and children have low trust level. Principals also have a dynamic attribute *location*: its value corresponds to the principal's location as defined by their mobile phone GPS. The conditions defining the three *Core* categories are:

$$\begin{aligned} LT.cond &= (p.level = low\ \mathsf{and}\ p.location = atHome)\\ MT.cond &= (p.level = medium\ \mathsf{and}\ p.location = atHome)\\ FT.cond &= (p.level = full) \end{aligned}$$

In *Core*, there is a hierarchical relation between categories: $FT \subseteq MT \subseteq LT$. Intuitively, LT is very restrictive (i.e., there are very few tuples in $\mathcal{ARCA}$ for LT), MT involves some restrictions and principals in FT will be granted full permissions (FT inherits the permissions of MT, which inherits those of LT).

Extended template. The three *Core* categories can be refined: sub-categories can be defined by using environmental attributes in $\mathcal{EAtA}$. We consider an extended template with the following hierarchy of principal categories:

$$FT \subseteq FMT \subseteq MT \subseteq Supervised \subseteq LT$$

The principal category $Supervised \subseteq LT$ is predefined as:

$$Supervised.cond = (p.level = low \text{ and } p.location = atHome \text{ and } env.owner@Home)$$

Thus *Supervised* inherits all the permissions of LT according to the CBAC axiom ($a1$) and it could have more permissions. For example a child in the *Supervised* category might be able to access the TV even if the LT category does not have this permission.

Similarly, a category $FMT \subseteq MT$ is included with the following condition:

$$FMT.cond = (p.level = medium \text{ and } p.location = atHome \text{ and } env.holiday)$$

Thus, FMT inherits the permissions of MT and can have more permissions. In this way, permissions can be delegated from the FT category to FMT household members during holidays.

Summarising, *Core* includes the following pre-defined categorisation for principals:

$$\begin{aligned}\mathcal{PCA}(p) = {} & \text{if } p.level = low \text{ and } p.location = atHome \text{ then} \\ & (\text{if } env.owner@Home \text{ then } Supervised \text{ else } LT) \\ & \text{else} \\ & \text{if } p.level = medium \text{ and } p.location = atHome \text{ then} \\ & (\text{if } env.holiday \text{ then } FMT \text{ else } MT) \\ & \text{else} \\ & \text{if } p.level = full \text{ then } FT \\ & \text{else } error\end{aligned}$$

For users with a more technical background or in cases where more fine-grained policies are required (for example, families that prefer to differentiate young children from teenagers), the full *SHoCBAC* model can be used.

For devices, categories in *Core* are associated with the device *type* (an attribute stored in $\mathcal{RAtA}$) and are also hierarchically organised. The main device types are *Cooking* (e.g., smart oven), *Heating*, *Entertainment* (e.g., TV, playstation), *Lights*, *Security* (including devices such as alarm and door lock) and *Cameras*. The user can add, delete or refine these categories. Any number of devices can be assigned to a category, however, all the devices in a category must have the same actions.

Category refinement. To refine the above-mentioned static device categories, environmental attributes can be used (e.g., date, time, weekDay/weekEnd, dayTime/nightTime). For instance, we may need to specify a category ensuring that unsupervised children cannot access the TV. In *Core*, the category $EntRestricted \subseteq Entertainment$ can be used to restrict access to entertainment devices. The idea is that $\mathcal{ARCA}$ includes permissions for principals in LT to access devices in the category $Entertainment$, but devices in $EntRestricted$ can only be accessed by principals in $Supervised$. The hierarchical relation ensures that unsupervised children (in the category LT) cannot access the TV. Similarly, the device category Security can be restricted during the weekend:

$$\begin{aligned} EntRestricted.cond &= (r.type = Entertainment \ \mathsf{and}\ env.date \in env.weekDay \\ &\qquad \mathsf{and}\ env.time \in env.nightTime) \\ SecRestricted.cond &= (r.type = Security \ \mathsf{and}\ env.date \in env.weekEnd) \end{aligned}$$

Given the above definitions, the category of a device can be directly computed as specified in the C-ABAC axioms (i.e., by evaluating the conditions using the attribute values at the point of the access request). For example:

$$\begin{aligned} \mathcal{RCA}(r) = {} & \mathsf{if}\ r.type = Entertainment\ \mathsf{then} \\ & (\mathsf{if}\ env.date \in env.weekDay\ \mathsf{and}\ env.time \in env.nightTime \\ & \mathsf{then}\ EntRestricted\ \mathsf{else}\ Entertainment) \\ & \mathsf{else} \\ & \mathsf{if}\ r.type = Security\ \mathsf{then} \\ & (\mathsf{if}\ env.date \in env.weekEnd\ \mathsf{then}\ SecRestricted\ \mathsf{else}\ Security) \\ & \ldots \end{aligned}$$

There are two categories of actions for each category of device, namely *Safe* and *Unsafe*, as well as a category *All* which contains the union. For example, the actions On and changeTemperature are in *UnsafeCooking*, whereas Off is in *SafeCooking*. *Core* includes the assignment of safe and unsafe actions to categories of principals as follows: only safe actions in the categories Entertainment, Lights and Cooking are assigned to LT; Supervised and MT include all safe and unsafe actions in those categories, as well as safe and unsafe actions on the Heating and Security categories, the only restriction is the actions on Cameras (e.g., we may not want the house cleaner in MT to manipulate cameras, however this may be allowed for the house keeper in FMT); FT is the top of the hierarchy and has no restrictions. The tuples defining the core policy as well as the more refined policy, where the categories Supervised, FMT, EntRestricted and SecRestricted could be used, are synthesised in Table 1. More general policies, where the categorisation of actions may depend for example on environmental attributes, could be defined using the general model.

Summarising, the administrator simply needs to assign a trust level (Full, Low or Medium) to each principal and a type to each device (Cooking, Entertainment, Security, etc.), then (optionally, if using the extended core template) parameterise environmental attributes such as holiday dates or Day/Night time and indicate whether any of the levels/types should be refined and whether the

Table 1. *SHoCBAC* policy templates

Minimal policy
Principal categories: LT, MT, FT.
Device categories: $Cooking, Lights, Entertainment, Heating, Security, Cameras$.
Permissions in $\mathcal{ARCA}$: (SafeCooking, Cooking, LT), (SafeLights, Lights, LT), (All, Entertainment, LT), (All, Heating,MT), (All,Security, MT), (All,Cameras, FT).
Extended policy
Principal categories: $LT, Supervised, MT, FMT, FT$
Device categories: minimal categories plus EntRestricted, SecRestricted.
Permissions in $\mathcal{ARCA}$: *minimal policy permissions,* (UnsafeCooking, Cooking, Supervised), (All, EntRestricted, Supervised), (All, SecRestricted, MT), (All,Security, FMT), (All,Cameras, FT).

standard permissions associated to principal categories are appropriate, changing any assignments if required. The authorisations are then automatically derived.

Core includes an administrative level that contains only one category, Owner-Admin, the only category with access to the log files keeping track of the actions that have been carried out on all the devices.

Policy analysis techniques developed for CBAC are directly applicable to *SHoCBAC*, see for example [14, 12]).

3.3 Smart Home Policies in *Core*

In this section we consider some standard rules that user-studies [17] suggest for Smart Home IoT access control, and show how they can be modelled using *Core*.

- *Children should never be able to use certain capabilities without supervision.* To model this rule, it is sufficient to give children a low trust level. Then they will be categorised in LT, which does not have access to unsafe operations on devices. If using the Supervised category, they will automatically be assigned to *Supervised* when the parents are at home. For example, assume Chloe is a child and Chloe.level = low in $\mathcal{PAtA}$. Her category is computed using the following rule:

$$\mathcal{PCA}(Chloe) = \mathsf{if}\ Chloe.level = low\ \mathsf{and}\ Chloe.location = atHome \\ \mathsf{then\ if}\ env.owner@Home\ \mathsf{then}\ Supervised\ \mathsf{else}\ LT$$

- *Babysitters, neighbours, and visitors should only be able to use any capabilities while in the house.* Assuming Babysitters, neighbours and visitors are assigned to the medium trust level, the condition $p.location = atHome$ in the definition of $\mathcal{PCA}$ ensures that they only have access to the devices while they are at home.

- *Any user who is currently at home should always be allowed to adjust lighting.*
  By assigning the actions to adjust lighting devices to the SafeLights category, we ensure that anyone at home can adjust lighting, since we have a tuple *(SafeLights, Lights, LT)* and all users belong to LT when they are at home. The latter is a consequence of the condition $p.location = atHome$ in the $\mathcal{PCA}$ rule that assigns the category LT to principals with low trust level, and the fact that $FT \subseteq MT \subseteq LT$, i.e., principals in the categories FT and MT inherit the permissions from LT.
- *Entertainment devices can be used freely by teenagers during the weekend, but need parental supervision on weekdays.*
  To specify this restriction for teenagers, we need a category with less permissions than MT but more than Supervised: If we assign teenagers medium trust level, then to ensure this restriction we will need to restrict all MT members, which might not be what is intended. If we assign teenagers low trust level then they will also need parental supervision on weekends (the Supervised category does not distinguish weekends from week days). This example shows the limits of *Core*, which is based on trust levels rather than specific characteristics of the principals. This restriction can be specified instead in the general *SHoCBAC* model.
- *Neighbours can have some additional capabilities when the owner is on vacation, e.g. the possibility to turn off the alarm.*
  By assigning neighbours medium trust level, they will be in the MT category if the owners are not on holidays, and in the FMT category when the owners are on holidays, which will give them the capability to perform all actions on security devices.
- *No one should be allowed to delete log files.*
  Since the action to delete log files has not been assigned to any category of users, no user is allowed to delete log files.
- *Spouses should have access to all capabilities, except for deleting log files.*
  It is sufficient to assign spouses the full trust level, which will categorise them as FT. All the capabilities are then available.

Core is sufficiently flexible to accommodate standard policies. As in RBAC, we can define static categories using trust levels, and as in ABAC, category membership can be refined dynamically without the intervention of an administrator. However, there may be situations where more involved policies are needed. This can be done by using the full power of *SHoCBAC*.

4 Expressive Power

In this section we show how to specify HyBAC policies [4] in *SHoCBAC*. Due to space limitations, we focus on $HyBAC_{RC}$, which follows a role-centric approach, and leave out the encoding of the attribute-centric $HyBAC_{AC}$ (which is closer to C-ABAC, and thus to *SHoCBAC*). As mentioned earlier, we do not consider sessions in this paper. If needed they can be represented as attributes of principals.

We start by briefly recalling the $HyBAC_{RC}$ model. We show next how CBAC policies can mimic $HyBAC_{RC}$ policies and produce the same result, while being more concise. The $HyBAC_{RC}$ definitions and examples are taken from [4].

$HyBAC_{RC}$ policy. A $HyBAC_{RC}$ policy is composed of

- **users, roles and their relations**. Specifically, finite sets U of users and R of roles, and the user-role assignment relation $UA \subseteq U \times R$.
- **devices, operations and their relations**. Specifically, finite sets D of devices, DR of device roles, OP of operations, and a device-operation assignment relation $P \subseteq D \times OP$. A pair $(d \in D, op \in OP) \in P$ means that operation op is permitted on device d. For instance we may have $(Oven, Open_{oven}) \in P$. The relation PDRA is a many-to-many relation assigning permissions to device roles: $PDRA \subseteq P \times DR$. For instance, $((Oven, Open_{oven}),$ $Dangerous\_kitchen\_perm) \in PDRA$. For all permissions p, we denote $droles(p) = \{dr \mid (p, dr) \in PDRA\}$.
- **environment related information**. There is a finite set ER of environment roles, triggered by a set EC of environment conditions determined by sensors (e.g., in the kitchen, evening). They are related by a many-to-many environment role activation relation $EA \subseteq 2^{EC} \times ER$. For instance we may have $((weekend, evening), Kids\_Entertainment\_time) \in EA$.
- **dynamic attributes**. Specifically, finite sets of attributes associated with users ($DUSA$) and with devices (DDA), such that $DUSA \cap DDA = \emptyset$.
- **role pairs and their relations**. There is a relation $RP \subseteq R \times 2^{ER}$ pairing user roles and environment roles (eg. $(kids, Kids\_Entertainment\_time) \in RP$) and derived relations $RPRA \subseteq RP \times R$ and $RPEA \subseteq RP \times 2^{ER}$.
  The role-pair assignment relation $RPDRA \subseteq RP \times DR$ binds all these components together: for any role $rp \in RP$, the role r associated with it through $RPRA$ has access to all device roles assigned to it through $RPRA$, when the set of environment roles associated with it through $RPEA$ are active. Note that actions are not mentioned explicitly in this relation, the user roles are linked with the device role which forces one to define different device roles, corresponding to the different capabilities of the device.

Evaluation of an access request. In $HyBAC_{RC}$, $CheckAccess(u, op, d, ec)$ is evaluated when a user u asks to perform an operation op on a device d when the environment condition $ec \in EC$ is active. The $CheckAccess$ predicate is composed of an authorisation function $Authorization(u, op, d)$ which evaluates to $True$ if u is allowed to perform op on d according to the current attributes values of u and d. The $CheckAccess$ predicate also includes a formula verifying

- role membership, i.e. there is an assignment role pair $(rp, dr) \in RPDRA$ such that dr is assigned the permission $(d, op) \in PDRA$
- role activation requirements: each environment role er in the role pair $rp = (r, ER_{rp})$ is activated by the currently active environment conditions ec.

Encoding. $HyBAC_{RC}$ policies are translated into *SHoCBAC* as follows:

- **users, roles and their relations**. The set U of users is represented by the set $\mathcal{P}$ of principals in *SHoCBAC*, static attributes for users correspond to user categories $\mathcal{C}_P$ and their assignments $(u, a) \in UA$ are encoded by the relation $\mathcal{PCA}$.
- **devices, operations and their relations**. The set D of devices is represented by the set $\mathcal{R}$ of resources, device roles are resource categories $\mathcal{C}_R$ in *SHoCBAC* and their assignments DR are encoded in the relation $\mathcal{RCA}$. Device operations OP are the actions $\mathcal{A}$ in *SHoCBAC* and permissions P are assigned via the relation $\mathcal{ARCA}$ (we assume here each category action corresponds to a single action in $\mathcal{ACA}$). We don't need to create the relation P, as it generates redundancies. We assign via $\mathcal{ARCA}$ the actions to categories of resources instead of linking them to each resource. For instance, if $((Oven, Open_{oven}), Dangerous\_kitchen\_perm) \in PDRA$, then we may have $(Adult, Open, Dangerous\_kitchen\_perm) \in \mathcal{ARCA}$, with $Oven \in Dangerous\_kitchen\_perm \in \mathcal{RCA}$. More generally, for all $d \in D$, if $(d, op) \in P$ and $roles(d) = dr$ then dr is translated into a resource category c_r, *op* is translated into an action a to which the category c_r is related via the permission relation $\mathcal{ARCA}$.
- **environment related information**. Environmental conditions (possibly using sensors information) are built from environmental attributes in $\mathcal{EAtA}$ and are included in category-attribute assignment conditions in $\mathcal{CAtA}$. Environmental roles may thus be seen as principal or ressources (sub-)categories. Role activation is automatically ensured by (dynamic) category assignment $\mathcal{CAtA}$ in axioms (c1) and (c2).
  For instance $Kids\_Entertainment\_time \in ER$ can be seen as a subcategory of $Entertainment\_devices \in C_R$ when the condition *env.Evening and env.weekEnd* is satisfied (thus including the environmental attributes in EA, directly in the dynamic definition of ER, see Section 3.2).
- **dynamic attributes**. Attributes are assigned to users and resources using the relations $\mathcal{PAtA}$ and $\mathcal{RAtA}$, respectively.
- **role pairs and their relations**. We do not need a specific relation to pair environmental and user roles. They correspond to principal or device categories and are linked (with the corresponding actions) using the $\mathcal{ARCA}$ relation and using the hierarchical relation $\subseteq$ between categories.
  For instance, consider the pair
  $((kids, Kids\_Entertainment\_time), Entertainment\_devices)) \in RPDRA$,
  where $Entertainment\_devices$ is the role associated to the resources {TV, DVD, PlayStation} and possible actions are {On, Off}. This shows that kids can access *Entertainment Devices* device role when the environment conditions *Kids Entertainment Time* is active.
  This is encoded in *SHoCBAC* with two categories $Kids\_Entertainment$ and $Entertainment\_devices$ such that $Entertainment\_devices \subseteq Kids\_Entertainment$. Both categories contain the resources {TV, DVD, PlayStation}. Assume there is a category of actions *Safe* containing On and

Off. To restrict kids access to Entertainment devices, we include only the tuple $(Safe, Kids\_Entertainment, kids)$ in $\mathcal{ARCA}$, whereas for other user categories we may have additional tuples granting access to Entertainment.

Proposition 1. *A user u is given permission to execute the operation op on a device d under certain environmental conditions ec in an $HyBAC_{RC}$ policy if and only if $(u, op, d) \in \mathcal{PAR}$ under the same environmental conditions in the translated SHoCBAC policy.*

Proof. (Sketch) $\Rightarrow$: A request by u to perform op on d is granted in $HyBAC_{RC}$ if $CheckAccess(s, op, d, ec)$ evaluates to $True$, that is, $Authorization(u, op, d)$ evaluates to $True$, i.e. the operation is allowed on the device for the user u considering the actual attribute values for the user, the device and their roles. In CBAC this is ensured by the fact that the $\mathcal{PCA}$ and $\mathcal{RCA}$ relations used to derive $\mathcal{PAR}$ are defined using the principal, resource (and environmental) attributes (see axioms c1 and c2). The user/device role is the principal/resource's category given by $\mathcal{PCA}$ ($\mathcal{RCA}$ respectively), and the permitted actions, i.e., the PDRA relation, are encoded in the $\mathcal{ARCA}$ relation.

Concerning the roles activation requirements, this is ensured by the definition of the relation $\mathcal{CAtA} \subseteq \mathcal{C} \times \mathcal{C}ond$, which includes in its condition environmental attributes (relation $\mathcal{EAtA}$). This means that by definition a category (role) cannot be assigned in CBAC (i.e. activated) if the actual values of the environmental attributes are not satisfied (see axioms $c1$ and $c2$).

Role membership is ensured by construction. As explained, environmental conditions are included in the definition of principal or resource categories and the related permissions are given by the relation $\mathcal{ARCA}$, which is derived from $PDRA$.

$\Leftarrow$: If $\mathcal{PAR}(u, op, d)$ is granted in *SHoCBAC*, this means that there is a resource category $c_r \ni d$, a principal category $c_p \ni u$ and an action category $c_a \ni op$ such that $(c_a, c_r, c_p) \in \mathcal{ARCA}$. We need to define in the $HyBAC_{RC}$ policy a device role $c_r \in DR$, include $(d, op) \in P$ and include $((d, op), c_r) \in PDRA$. If environmental attributes in $\mathcal{EAtA}$ are used in the condition defining c_r (or c_p), then the corresponding environmental role $er \in ER$ needs to be defined and paired with the device role c_r, i.e. $((c_p, er), c_r) \in RPDRA$. The function $Authorization(s, op, d)$ is defined as $c_p \in roles(u) \wedge c_r \in droles((op, d))$. □

The $HyBAC_{RC}$ model also includes some constraints such as PRConstraints, which prevent RPDRA assignments that would enable specific roles to access specifically prohibited permissions; Static Separation of Duty (SSD) and Dynamic Separation of Duty (DSD), which are the usual notions used e.g. in RBAC. PRConstraints can be modelled in CBAC using the relation $\mathcal{BARCA}$ (see [10]). SSD (as well as binding of duty constraints) can be specified by adding axioms at the metamodel level (see [11]). Verifying DSD can be done similarly to $HyBAC_{RC}$, if sessions are introduced.

The encoding of $HyBAC_{AC}$ is similar, but more direct since both $HyBAC_{AC}$ and *SHoCBAC* are ABAC-style models.

5 Related Work

A variety of access control models have been proposed for the Internet of Things (e.g., [4, 20, 8, 5, 16]). Here we discuss the models that are closer to ours.

Following Kuhn et al [18] RBAC and ABAC combinations can be classified as Attribute-centric, Role-centric, or using Dynamic roles. C-ABAC is in the third class: it is a dynamic version of RBAC, where roles (represented by categories), are defined by Boolean formulas on attribute values, and the same idea of categorisation is applied also to resources. Ameer et al. [4] follow the attribute-centric and role-centric approaches to define smart home access control models. Instead, here we follow the dynamic role approach, which was not considered in previous work due to the potentially large number of attributes in smart homes. However, even if smart home applications are rich in attributes, user studies [17] show that there is only a small set of meaningful attribute combinations, and CBAC offers a natural way to use them to define authorisations.

SHoCBAC is able to express the policies defined in previous models, such as $HyBAC_{RC}$ and $HyBAC_{AC}$ (which were shown to generalise EGRBAC [2] and HABAC [3]) and can also express other models in between these two. $HyBAC_{RC}$ can specify role constraints to be checked at configuration time, whereas $HyBAC_{AC}$ can specify role constraints that are checked at execution time. The flexibility of CBAC allows us to define instances with both kinds of constraints, i.e., constraints checked at execution or configuration time.

Note that HABAC has been shown in previous work to be unable to express role constraints as defined in EGRBAC. Despite being an ABAC-based model, *SHoCBAC* is able to express EGRBAC policies, including role constraints. This can be done using the administrative level (which is also an instance of the CBAC metamodel), i.e., we can ensure administrators are not allowed to include an $\mathcal{ARCA}$ tuple if there is a $\mathcal{BARCA}$ one. This would mimic EGRBAC behaviour.

While previous access control models for the Smart Home IoT need a separate administrative model, see for example [21], *SHoCBAC* allows also the specification of administrative authorisations and prohibitions in a uniform way. In future work we will provide a detailed analysis of administration in *SHoCBAC*.

6 Conclusions and Future Work

SHoCBAC is a logic-based access control model for smart homes, which uses categorisation (a key mechanism underlying cognition). It is sufficiently expressive to satisfy the requirements of smart homes, where policies should be fine-grained and dynamic. From a usability point of view, we have also identified an instance of *SHoCBAC* based on trust levels, which is simple yet covers the standard Smart Home IoT requirements. *SHoCBAC* and its instances inherit the logic foundations of CBAC and its rewriting operational semantics, which provides a basis for the analysis of policy properties. Administrative policies can also be defined by using the CBAC administrative model. This will be the subject of future work.

References

1. S. Alves and M. Fernández. A graph-based framework for the analysis of access control policies. *Theor. Comput. Sci.*, 685:3–22, 2017.
2. Safwa Ameer, James O. Benson, and Ravi S. Sandhu. The EGRBAC model for smart home iot. In *21st International Conference on Information Reuse and Integration for Data Science, IRI 2020, Las Vegas, NV, USA, August 11-13, 2020*, pages 457–462. IEEE, 2020.
3. Safwa Ameer, James O. Benson, and Ravi S. Sandhu. An attribute-based approach toward a secured smart-home iot access control and a comparison with a role-based approach. *Inf.*, 13(2):60, 2022.
4. Safwa Ameer, James O. Benson, and Ravi S. Sandhu. Hybrid approaches (ABAC and RBAC) toward secure access control in smart home iot. *IEEE Trans. Dependable Secur. Comput.*, 20(5):4032–4051, 2023.
5. Safwa Ameer, Ram Krishnan, Ravi S. Sandhu, and Maanak Gupta. Utilizing the DLBAC approach toward a ZT score-based authorization for iot systems. In Mohamed Shehab, Maribel Fernández, and Ninghui Li, editors, *Proceedings of the Thirteenth ACM Conference on Data and Application Security and Privacy, CODASPY 2023, Charlotte, NC, USA, April 24-26, 2023*, pages 283–285. ACM, 2023.
6. M. Augoustinos, I. Walker, and N. Donaghue. *Social cognition: An integrated introduction.* Sage, 2014.
7. Steve Barker. The next 700 access control models or a unifying meta-model? In *SACMAT 2009, 14th ACM Symposium on Access Control Models and Technologies, Stresa, Italy, June 3-5, 2009, Proceedings*, pages 187–196, New York, 2009. ACM Press.
8. Elisa Bertino, Ravi S. Sandhu, Bhavani Thuraisingham, Indrakshi Ray, Wenjia Li, Maanak Gupta, and Sudip Mittal. Security and privacy for emerging IoT and CPS domains. In Anupam Joshi, Maribel Fernández, and Rakesh M. Verma, editors, *CODASPY '22: Twelveth ACM Conference on Data and Application Security and Privacy, Baltimore, MD, USA, April 24 - 27, 2022*, pages 336–337. ACM, 2022.
9. Clara Bertolissi and Maribel Fernández. Category-based authorisation models: operational semantics and expressive power. In *Proc. of Int. Symposium on Engineering Secure Software and Systems, ESSOS 2010, Pisa*, number 5965 in Lecture Notes in Computer Science, pages 140–156, Berlin, Heidelberg, 2010. Springer.
10. Clara Bertolissi and Maribel Fernández. A metamodel of access control for distributed environments: Applications and properties. *Inf. Comput.*, 238:187–207, 2014.
11. Clara Bertolissi, Maribel Fernández, and Bhavani Thuraisingham. Admin-CBAC: An administration model for category-based access control. In Vassil Roussev, Bhavani Thuraisingham, Barbara Carminati, and Murat Kantarcioglu, editors, *CODASPY '20: Tenth ACM Conference on Data and Application Security and Privacy, New Orleans, LA, USA, March 16-18, 2020*, pages 73–84. ACM, 2020.
12. Clara Bertolissi, Maribel Fernández, and Bhavani Thuraisingham. Graph-based specification of Admin-CBAC policies. In Anupam Joshi, Barbara Carminati, and Rakesh M. Verma, editors, *CODASPY '21: Eleventh ACM Conference on Data and Application Security and Privacy, Virtual Event, USA, April 26-28, 2021*, pages 173–184. ACM, 2021.
13. M.W. Eysenck and M. Brysbaert. *Fundamentals of cognition.* Routledge, 2018.
14. Maribel Fernández, Ian Mackie, and Bhavani M. Thuraisingham. Specification and analysis of ABAC policies via the category-based metamodel. In *Proceedings of the*

Ninth ACM Conference on Data and Application Security and Privacy, CODASPY 2019, Richardson, TX, USA, March 25-27, 2019, pages 173–184, New York, 2019. ACM Press.

15. Maribel Fernández, Jenjira Jaimunk, and Bhavani Thuraisingham. A privacy-preserving architecture and data-sharing model for cloud-iot applications. *IEEE Transactions on Dependable and Secure Computing*, 20(4):3495–3507, 2023.
16. Maanak Gupta, Smriti Bhatt, Asma Hassan Alshehri, and Ravi S. Sandhu. *Access Control Models and Architectures For IoT and Cyber Physical Systems.* Springer, 2022.
17. Weijia He, Maximilian Golla, Roshni Padhi, Jordan Ofek, Markus Dürmuth, Earlence Fernandes, and Blase Ur. Rethinking access control and authentication for the home internet of things (iot). In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 255–272. USENIX Association, 2018.
18. D. Richard Kuhn, Edward J. Coyne, and Timothy R. Weil. Adding attributes to role-based access control. *IEEE Computer*, 43(6):79–81, 2010.
19. Denis Obrezkov and Karsten Sohr. UCAT: the uniform categorization for access control. In Mohamed Mosbah, Florence Sèdes, Nadia Tawbi, Toufik Ahmed, Nora Boulahia-Cuppens, and Joaquín García-Alfaro, editors, *Foundations and Practice of Security - 16th International Symposium, FPS 2023, Bordeaux, France, December 11-13, 2023, Revised Selected Papers, Part II*, volume 14552 of *Lecture Notes in Computer Science*, pages 3–14. Springer, 2023.
20. Indrakshi Ray, Ramadan Abdunabi, and Rejina Basnet. Access control for internet of things applications. In *Proceedings of the 5th ACM on Cyber-Physical System Security Workshop, CPSS@AsiaCCS 2019, Auckland, New Zealand, 8 July 2019*, pages 35–36. ACM, 2019.
21. Mehrnoosh Shakarami and Ravi S. Sandhu. Role-based administration of role-based smart home iot. In Maanak Gupta, Mahmoud Abdelsalam, and Sudip Mittal, editors, *SAT-CPS@CODASPY 2021, Proceedings of the 2021 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems, Virtual Event, USA, April 28, 2021*, pages 49–58. ACM, 2021.