DRAFT: NOT FOR GENERAL DISTRIBUTION
The EGRBAC Model for Smart Home IoT Access Control

Safwa Ameer
The University of Texas at San Antonio

James Benson
The University of Texas at San Antonio

Ravi Sandhu
The University of Texas at San Antonio

Abstract

Recently, the concept of IoT (Internet of Things) has gained tremendous attention in both research and industry. In the near future IoT will affect all industries and everyone's daily life. In particular, it will be part of every home turning our houses into smart houses, in which we have multiple users with complex social relationships between them using the same smart devices. This requires usable authentication and sophisticated access control specification mechanisms that are currently lacking. In this paper, we introduce the extended generalized role based access control (EGRBAC) model for smart home IoT. EGRBAC is a dynamic and fine-grained model, suitable for constrained home environments. We provide a formal definition of the model and illustrate its features by several use case scenarios. We further provide an analysis of the beneficial attributes of EGRBAC as well as its limitations. Finally, we provide a proof-of-concept implementation for a consolidated use case in Amazon Web Services (AWS) IoT platform, followed by discussion of future enhancements. We envisage EGRBAC as the first step in developing a family of access control models for smart home IoT ranging from relatively simple and complete to incorporating increasingly sophisticated and comprehensive features.

Keywords

IoT, smart homes, access control, RBAC

1 Introduction and Motivation

The Internet of Things (IoT), sometimes called the Internet of Everything or the Industrial Internet, is a new technology paradigm envisioned as a global network of machines and devices capable of interacting with each other [36]. Currently, IoT is one of the most talked about topics in technology. It has already become an indispensable component of our lives. One of the most popular domains for deploying smart connected devices is the smart home.

To date, IoT security and privacy research has focused on such devices' insecure software-engineering practices, improper information flows, and the inherent difficulties of patching networked devices. Surprisingly little attention has been paid to access control policy specification, or authentication in home IoT [26]. Authorization issues have been explored extensively for many different domains. However, home IoT is significantly different from traditional domains such as enterprise, electronic commerce and web services in three main ways. First, in home IoT we have many users who use the same device, for example a smart door lock. Widely deployed techniques for specifying access-control policies and authenticating users fall short when multiple users share a device [26]. Second, house residents usually have complex social relationship between them, which introduce a new threat model, e.g. a annoying child trying to control the smart light in a sibling's room, a current or ex-partner trying to abuse one or all house residents [26, 39]. Another major characteristic of IoT devices is that the majority do not have screen and keyboards making them hands free for convenience while making authentication and access control more challenging. The characteristics that make IoT distinct from prior computing domains necessitate a rethinking of access control and authentication [26]. In particular, the need arises for a dynamic and fine-grained access control mechanism, where users and resources are constrained [46]. Real world examples of the shortcomings of current access control policy specification and authentication for home IoT devices have begun to appear as described in [26], [54], and [27]. In this research, we studied the literature IoT access control models, analyzed it, and came up with criteria that need to be satisfied in smart home access control models. Moreover, for IoT access control models that govern user to device access, we investigated them against our criteria, and notably no model satisfies all desired specifications.

Our goal is to address the lack of widely-adopted access control models for smart home IoT. We believe the best approach for this purpose is to develop a family (or series) of models ranging from relatively simple and complete to incorporating increasingly sophisticated and comprehensive features. Developing such a family has been successful in the past, most notably in the seminal role-based access control (RBAC) models of [21, 51]. Other access control model families have been published in a variety of contexts including usage control [48], role-based delegation [8], on-line social networks [12, 22], attribute-based access control (ABAC) [31] and relationship-based access control [3]. The complexities of the smart home IoT environment similarly merits development of a suitable family of access control models.

In this paper we describe our first access control model for smart home IoT, which will serve as a foundation for eventual development of a larger family of models. Here, one might argue that why just the home environment, how about the general IoT context? As we mentioned earlier in this section, smart homes have unique characteristics which requires a special access control model. However, this model can be altered, and adjusted to conform with the access control specifications of other IoT domains. Our model is inspired by the early work of Covington et al [14] which, rather informally, presented the generalized role-based access control (GRBAC) model for aware homes (pre-dating current terminology of smart homes). Our model called the extended generalized role-based access control (EGRBAC) model builds upon GRBAC. GRBAC introduced the crucial concepts of environment and device roles which are also central to EGRBAC. Building upon GRBAC maintains continuity

with the pre-IoT literature, and begins with the simpler concepts of RBAC as opposed to the more sophisticated notions of ABAC. Similar to GRBAC, EGRBAC also focusses on user to device (U-D) interaction while leaving device to device (D-D) interaction out of scope. More complete and comprehensive models built around EGRBAC would need to account for D-D interactions, as well as bring in elements of ABAC. A major benefit of ABAC over RBAC, is its ability to capture contextual attributes (such as time, location, and home occupied/not occupied status). EGRBAC accommodates this ability via environment roles (adapted from GRBAC), whereby ABAC enhancements to EGRBAC would need to demonstrate improvement on environment roles as well as bring in additional ABAC aspects.

The paper is organized as follows. Section 2 identifies desirable criteria for smart home IoT access control models. An analysis and review of related work is given in Section 3. Section 4 provides an overview of GRBAC [14], and of the architecture that we adopt to enforce EGRBAC in our proof-of-concept. Section 5 describes our threat model. Our formalization of GRBAC model is explained in Section 6. In Section 7, we introduce the EGRBAC model along with two use case scenarios. An analysis of the benefits and limitations of EGRBAC is provided in Section 8. A proof-of-concept demonstration and associated performance analysis are discussed in Section 9. Section 10 concludes the paper.

2 Criteria for Smart Home IoT Access Control Models

Ouaddah et al [46] have provided a survey on IoT access control models, identified the main challenges, and potential future directions. However, they only discussed the advantages and disadvantages of each model, and did not investigate all IoT access control's model against a unified criteria. He et al [26] have recently proposed a new perspective of access control policies specifications for home IoT. They identify four access control policy characteristics that need to be maintained in smart homes, as follows. (i) Access control should be fine-grained at the level of individual operations on devices (called capabilities in [26]) rather than at the device level. (ii) The complex social relationships between the house members play an important role in access control policies. (iii) Smart home IoT access control policies are highly impacted by contextual factors. (iv) There are some commonly occurring preferences amongst home users that can be configured as a default setting.

Here, we introduce our specifications for smart home IoT access control models. The first two characteristics are inspired by He et al [26] perspective. We believe that a smart home IoT access control model (whether it is device to device (D-D), user to device (U-D) or both) should exhibit, at least, the following characteristics. (i) The model should be dynamic so as to capture environment and object contextual information, as in He et al's third characteristic. (ii) The model should be fine-grained so that a subset of the functionality of a device can be authorized rather than all-or-nothing access to the device. This is similar to He et al's first characteristic. (iii) Smart things in homes are usually limited in term of computational power, and storage. Accordingly, the model should be suitable for constrained home environment, and should not require extensive computation or communication on the part of resource constrained devices. Furthermore, any access control solution for smart home IoT should consider the fact that a generic interoperability standard among IoT devices is still missing. (iv) The model should be constructed specifically for smart home IoT, or otherwise be interpreted for the smart home domain such as by appropriate use cases. To ensure that the model is suitable for smart home different specifications such as, social relationships between house members (similar to He et al's second characteristic), cost effectiveness, usability, and so on. (v) The model should be demonstrated in a proof-of-concept to be credible using commercially available technology with necessary enhancements. (vi) The model should have a formal definition, so that there is a precise and rigorous specification of the intended behavior.

3 Related Work

Smart home IoT has been extensively studied by security experts. Many researchers have focused on identifying IoT security and privacy vulnerabilities [5, 15, 17, 24, 44, 52, 55]. Moreover, to analyze IoT security challenges and security design issues in specific, many researchers have conducted studies of IoT frameworks (e.g. [19, 20, 28, 40, 44]). One of the critical security services in IoT that mostly all researchers agree upon is access control. Ouaddah et al [46] have extensively investigated access control in IoT environments. As discussed above, He et al [26] have recently proposed a new perspective of access control policies specifications for home IoT.

The rest of this section provides an analysis of IoT access control models from the literature based on the six characteristics identified in Section 2. The models are categorized according to their foundational model, viz., RBAC, ABAC, UCON and CapBAC. A summary of the analysis is provided in Table 1. In this table we only included access control models that govern user to device access, since this is the scope of our model. Models that only address device to device access are omitted from the table, but are discussed in the text. From the table we can notice that except for our model (summarized in the first row of the table), no model satisfies all desired characteristics. Furthermore, surprisingly, except for EGRBAC, and GRBAC no model was designed or interpreted explicitly for smart home environment. In Section 8 we justify the evaluation of EGRBAC according to the characteristics in this table.

IoT Access Control Models Based On RBAC

The basic concept of role based access control (RBAC) model [21, 50] is that permissions are associated with roles, and users are made members of appropriate roles, thereby acquiring the roles' permissions. Covington et al [14] developed GRBAC which introduced the notion of environment and device roles, to capture environmental conditions and to enable devices categorization respectively, but did not give a formal model. Subsequently they provided a brief but incomplete formalization without implementation [13]. Our EGRBAC model is inspired by GRBAC in part as will be discussed in Sections 6 and 7.

In [59] the authors extended RBAC by introducing context constraints. However, they mainly focused on the environment of web services. Researchers in [7, 32] proposed two different solutions, but both of them are focused on Web of Things [16, 53]. Their models are not adequate for smart homes. In the first solution, the architecture is completely centralized in a central access control decision

Table 1: Analysis of Published IoT Access Control Models Based on Desirable Characteristics

Model Type	Model	U-D or D-D	Dynamic	Fine Grained	Suitable for constrained home environment	Designed or interpreted for smart home IoT	Implemented	Provides a formal Access Control Model
RBAC Model	EGRBAC, this paper	U-D	yes	yes	yes	yes	yes	yes
RBAC Model	GRBAC, Covington et al [14]	U-D	yes	no	yes	yes	no	no
RBAC Model	Zhang et al [59]	U-D and D-D	yes	yes	yes	no	no	yes
RBAC Model	Barka et al [7]	U-D and D-D	no	yes	no	no	no	utilizes RBAC [51]
RBAC Model	Jindou et al [32]	U-D	no	yes	no	no	yes	yes
RBAC Model	Kaiwen et al [33]	U-D	yes	yes	yes	no	no	yes
RBAC Model	Liu et al [37]	U-D	no	yes	yes	no	no	no
ABAC Model	Ye et al [58]	U-D and D-D	yes	no	no	no	no	yes
ABAC Model	Bandara et al [6]	U-D	no	yes	yes	no	yes	utilizes XACML [49]
ABAC Model	Mutsvangwa et al [41]	U-D	N/A	N/A	no	no	no	no
ABAC Model	Xie et al [57]	U-D and D-D	N/A	N/A	no	no	no	no
UCON Model	Martinelli et al [38]	U-D	yes	yes	yes	no	yes	utilizes U-XACML [11, 35]
CapBAC Model	A survey is provided in [46]	Not adequate for the constrained environment of smart homes as explained in Section 3.						

facility coupled to a database, whereby access control decisions are taken outside the house requiring a live connection and increasing the attack surface. In our model decisions are made locally with diverse protocols. On the other hand, the main drawback of the second solution is the strong attachment to Social Network Services (SNS). Resource owners and requesters must have an SNS profile or account to interact with each other which is unsuitable in case of smart homes where we have kids that may not have a social network account, and we may have workers with whom one may not want to connect in social networks, like a plumber who should access the house for one time. Moreover, this solution introduces the SNS provider as a trusted third party. In [33] an Attribute-Role-Based Hybrid Access Control model was introduced for IoT in general and not specific to smart homes. Also, no implementation was provided. The RBAC model for IoT was also adopted in [37]. However, the authors focus on providing an authentication protocol, while they only gave a high level overview of their RBAC model.

IoT Access Control Models Based on ABAC

Different attribute-based access control (ABAC) models have been proposed in the literature (e.g. [29, 31]). In ABAC, access is granted according to attributes associated with the user and resource. In [9], the authors introduced an ABAC-based model that focuses on device to device access control. However, use cases and performance evaluation are lacking. Other access control models that are based on ABAC for IoT were proposed in [6, 41, 57, 58]. However, as observed by [4], it is not simple to design and implement an adequate ABAC model for IoT given that the implementation of ABAC usually requires heavy computation, which cannot be supported by constrained smart things. Furthermore, increasing the number of attributes can significantly increase the chance of conflict among the access policies, and it is not easy to detect and resolve these conflicts. Finally, identifying a set of sufficient attributes is critical, but also challenging. We should mention here that in [41], and [57] the authors focused on providing sophisticated attribute based encryption (ABE) models for smart grids, while they did not discuss the ABAC models that they consider. Moreover, an ABE model for smart grids may not be suitable for computationally constrained smart home things.

IoT Access Control Models Based on CapBAC

Capability-based access control (CapBAC) utilizes the concept of capability, first introduced in [30], as a token, ticket, or key that gives the possessor permission to access an entity or object in a computer system. Much work has been done in the literature using CapBAC in IoT. The major drawback in CapBAC model is that it requires that all devices must implement CapBAC, which is unlikely given the heterogeneity of a home smart things. Moreover, in CapBAC individual devices or gateways should act as policy decision points, which can be inconvenient on computationally

and power constrained devices. Authors in [46] gives a survey on solutions proposed using CapBAC model.

IoT Access Control Models Based on UCON

The distinguishing properties of usage control (UCON) beyond traditional ABAC are the continuity of access decisions and the mutability of subject and object attributes [47, 48, 60]. A few solutions have been proposed in the literature that are based on UCON. However, these models cannot be adopted yet for various reasons. In [25], the model is proposed as a Device to Services (D-S) access control model, moreover, no implementation was provided, instead, only two theoretical experiments were introduced and assessed. In [34], the authors mainly focused on providing a distributed Peer-to-Peer (P2P) architecture. They did not consider how to use their system to grant users access to different smart things in the house. In contrast, in our model we demonstrated by use cases study and implementation how to use our model to control users access to smart things. Finally, unlike our model, in [38], the authors did not consider justifying and illustrating the fitness of their model for smart home IoT access control challenges.

IoT Access Control Models Based on Blockchains

Some solutions built on blockchain technology have been proposed (e.g. [18, 43, 45]. However, as [43] described, the blockchain technology has some technical characteristics that could limit its applicability. First, cryptocurrency fees are typically a fundamental part of blockchain-based platforms. All the transactions include a fee, and miners are awarded with certain amount of cryptocurrency if they successfully manage to include one of their mined blocks into the blockchain. Second, processing time, transactions take time to get accepted into the blockchain. As of writing this paper, Bitcoin's [42] transactions can take up to 10 minutes and 12 seconds in Ethereum [56]. Because of these limitations we did not consider blockchain-based access control models in Table 1.

IoT Access Control Models Based on Other Models

In the literature, several other access control models for IoT have been proposed. The authors in [4, 46, 61] provide surveys on these. However, none of them provided an access control model that meet smart home IoT challenges and that is formalized, justified and implemented.

From the above analysis, as summarized in Table 1, we can conclude that none of the previously published models that have been proposed and extended from RBAC, ABAC, CapBAC, and UCON meet the specified desired characteristics.

4 Background

In this section we introduce GRBAC (Generalized Role Based Access Control) model [14]. EGRBAC is in part inspired by this model. We also present an IoT based smart home architecture [23], which we adopted to enforce EGRBAC.

The GRBAC Model

Covington et al introduced the Generalized Role-Based Access Control (GRBAC) model [14]. In addition to the usual concept of *Subject Role*, GRBAC incorporates the notion of *Object Roles* and *Environment Roles*. A subject role is analogous to a traditional RBAC role. An object role is defined as the properties of the resources in the system, such as images, source code, streaming videos, devices.

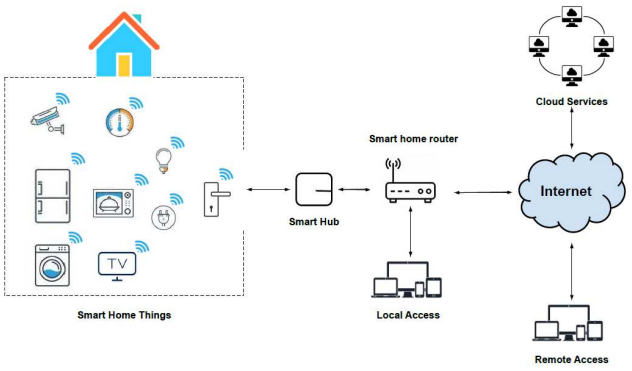

Figure 1: EGRBAC Enforcement Architecture (adapted from [23])

An nvironment role is defined as the environment state during access. Covington et al [13] subsequently described an architecture to support environment roles activation according to the current environment conditions. They also provided a high level but incomplete formal definition of environment role based access control model, building upon [50]. They neither considered formalizing the object role part of GRBAC, nor provided a model diagram. In Section 6, we provide a complete detailed formalization of GRBAC accompanied with a model diagram.

IOT Based Smart Home Architecture

The smart home IoT architecture that we adopted for EGRBAC enforcement was introduced by Geneiatakis et al [23]. It is illustrated in Figure 1. The IoT devices are connected to a corresponding hub and are not directly accessed by other devices or by users. This solves the problem of inter-operability among IoT devices. In addition, the intermediate hub is responsible for providing internet connectivity, since the majority of commercial sensors do not provide direct internet connectivity. The communication between the smart hub and the IoT devices is usually wireless, through different protocols such as Zigbee, Z-Wave and WiFi. In order to connect the smart IoT devices optionally, to the outside world, the hub is connected to the home's routers via an Ethernet or a Wi-Fi interface. The user can use different platforms to access the smart devices and manage the smart home, such as PCs, tablets and smart phones. In general there are two types of access. In local access users directly interact with the IoT devices through the connectivity services provided by the hub. In remote access users access IoT devices via cloud services, which in turn communicate with the smart hub via the Internet to access these devices.

5 Threat Model

In smart houses we recognize two types of adversaries [26]. First outsider hacker who is trying to get digital or physical access to the house by exploiting system vulnerabilities. Second the household members themselves, that is insiders who have legitimate digital and physical access to the house, such as family members, guests, and workers. The intention for legitimate user to break down the access control system of the smart home may vary from curiosity (e.g. a kid playing with oven setting), disturbing other family members

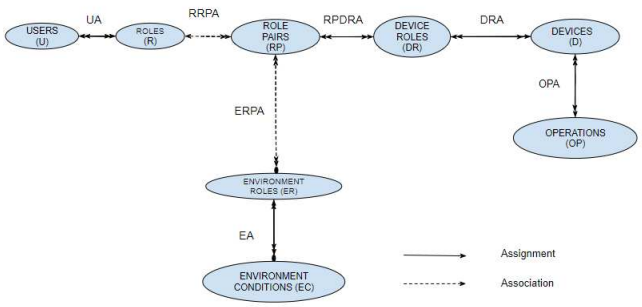

Figure 2: Our View of GRBAC Model

(e.g. a kid locking his brothers outside the house), to disobedience (e.g. a kid is trying to watch TV outside the allowed entertainment time), or robbery (e.g. a worker getting access to the camera system and adjust it to shutdown at a certain time). Making sure that those legitimate users get access only to what they are authorized to by the house owner, is the central focus of our paper. To enable this we need a suitable access control model conformant to the desired characteristics discussed in Section 2. We emphasize that authorized insiders who try to hack the access control system, or to break the IoT devices to get an unauthorized access to the system are outside the scope of our threat model.

6 GRBAC Model Formalization

In this section we develop a formal model for GRBAC. Since Covington et al [13, 14] only provide a partial formalization along with one simple use case, our formalization is a best effort at filling in missing details. It is a useful step in progressing towards EGRBAC.

Figure 2 depicts the components, i.e. the sets, relations and functions of GRBAC, and Table 2 formally defines these. Sets are shown as ovals in Figure 2, while the binary relations amongst them are shown as directed arrows with the single arrow indicating "one" and the double arrow "many." An arrow ending in a dot indicates a subset rather a single element of that set. A solid arrow represents assignment, whereas a dashed arrow indicates an association via mathematical definitions. Users (U) and Roles (R) are familiar sets in RBAC systems. A user is a human being who interacts with smart home devices as authorized. In context of smart homes, a role specifically represents the relationship between the user and the family, which encompasses parents, kids, neighbours, friends and such [26]. The many-to-many UA relation specifies the assignment of users to roles. A device (D) is a smart home device such as a smart TV. Operations (OP) represent actions on devices. Each device has a set of operations assigned by its manufacturer, represented as the many-to-many OPA relation. Device Roles (DR) are a mean of categorizing devices, such as entertainment, climate, or lighting. The many-to-many DRA relation specifies this assignment. Environment Roles (ER) are a GRBAC innovation representing environmental contexts, such as daytime/nighttime, weekday/weekend, and winter/summer. Environment roles are turned on/off (i.e., triggered) by Environment Conditions (EC) such as time, daylight, or weather. EA maps each environment role to multiple subsets

Table 2: GRBAC Model Formalization

Users, Roles and Devices
$-U, R, D, OP$ and DR are sets of users, roles, devices, operations and device roles respectively
$-UA \subseteq U \times R$, many to many users to role assignment (home owner specified)
$-OPA \subseteq OP \times D$, many to many assignment between operations and target devices (manufacturers specified)
$-DRA \subseteq D \times DR$, many to many devices to device roles assignment (home owner specified)

Environment Roles and Environment Conditions
$-ER$ and EC are sets of environment roles and environment conditions respectively
$-EA \subseteq 2^{EC} \times ER$, many to many subsets of environment conditions to environment roles assignment (home owner specified)

Role Pairs
$-RP \subseteq R \times 2^{ER}$, a set of role pairs specifying all permissible combinations of a user role and subsets of environment roles (home owner specified)
– For convenience for every $rp = (r_i, ER_j) \in RP$, let $rp.r = r_i$ and $rp.ER = ER_j$
$-RRPA \subseteq R \times RP$, one to many role to role pairs association, where $RRPA = \{(r_m, rp_n) \mid rp_n \in RP \wedge rp_n.r = r_m\}$
$-ERPA \subseteq ER \times RP$, many to many environment roles to role pairs association, where $ERPA = \{(er_m, rp_n) \mid rp_n \in RP \wedge er_m \in rp_n.ER\}$

Role Pair Assignment
$-RPDRA \subseteq RP \times DR$, many to many role pairs to device roles assignment (home owner specified)

Authorization Predicate
– For a user u_i to perform operation op_k on device d_j when the set of environment conditions EC_m is active:
$(op_k, d_j) \in OPA \wedge$
$(\exists\, r_x \in R, rp_y \in RP, dr_l \in DR)$, where:
$(u_i, r_x) \in UA \wedge rp_y.r = r_x \wedge (rp_y, dr_l) \in RPDRA \wedge$
$(d_j, dr_l) \in DRA \wedge$
$rp_y.ER \subseteq \{er \in ER \mid (\exists EC'_m \subseteq EC_m)[(EC'_m, er) \in EA]\}$

of EC. Suppose *Entertainment_Time* should be active on weekend evenings. We can use *weekends*, active during weekends, and *evenings*, active during evenings, and assign $(\{weekends, evenings\}, Entertainment\_Time)$ to EA. Each Role Pair (RP) is a combination of a role and currently active environment roles. A role pair rp has a role part $rp.r$ that is the single role associated with rp, and an environment role part $rp.ER$ that is the subset of environment roles associated with rp. The main idea of GRBAC, is that a user is assigned to one or more roles and according to the current active environment roles some role pairs will be active, so the user gets access to the devices assigned to the device roles assigned to the currently active role pairs. The permissible role pairs RP are specified as a subset of $R \times 2^{ER}$, since some ER subsets may not

$$U = \{alex, bob, \ldots\},\ R = \{kids, parents, \ldots\}$$
$$UA = \{(alex, kids), (bob, parents), \ldots\}$$
$$D = \{TV, DVD, Playstation, \ldots\}$$
$$DR = \{Entertainment\_Devices, \ldots\}$$
$$DRA = \{(TV, Entertainment\_Devices),$$
$$(DVD, Entertainment\_Devices),$$
$$(Playstation, Entertainment\_Devices), \ldots\}$$

$$EC = \{weekends, evenings, TRUE, \ldots\}$$
$$ER = \{Entertainment\_Time, Any\_Time, \ldots\}$$
$$EA = \{(\{weekends, evenings\}, Entertainment\_Time),$$
$$(TRUE, Any\_Time), \ldots\}$$

$$RP = \{(kids, \{Entertainment\_Time\}), (parents, \{Any\_Time\}), \ldots\}$$
$$RPDRA = \{((kids, \{Entertainment\_Time\}),$$
$$Entertainment\_Devices),$$
$$((parents, \{Any\_Time\}),$$
$$Entertainment\_Devices), \ldots\}$$

Figure 3: Use case 1.A configuration in GRBAC

be meaningful. Note that the *RRPA*, and *ERPA* relations, are determined by definition from *RP* and hence are associations rather than independent assignments. We should mention here that we introduced the role pair component to ease our formalization, and it has nothing to do with Covington et al [13] incomplete formalization. *RPDRA* brings all these components together by assigning device roles to role pairs. The authorization function of GRBAC is given at the bottom of Table 2. Consider a user u_i who attempts to perform operation op_k on device d_j when the subset of environment conditions EC_m is active. This operation will succeed if and only if all of the following are true: (i) There is an *OPA* relation between op_k and d_j. (ii) There exists a role r_x in *R*, a role pair rp_y in *RP*, and a device role dr_l in *DR*, where: (a) User u_i is assigned role r_x in *UA*. (b) The role part of rp_y is r_x. (c) There is a *RPDRA* relation between rp_y and dr_l. (d) There is a *DRA* relation between d_j and dr_l. (e) The environment role part of rp_y is a subset of the current active environment roles that have been triggered by the current active environment conditions. Following is a use case scenario that illustrates how to configure GRBAC to enforce specific access control policies.

Use Case Scenario 1.A The objective is to allow kids to use entertainment devices (TV, DVD, and PlayStation) during weekend evenings only, and allow parents to use entertainment devices unconditionally. GRBAC could be configured as shown in Figure 3 to achieve this objective. The ellipsis indicate that a partial specification is given since there may be additional users and devices, and additional policy elements that need to be specified. In Figure 3 there are two users *alex* and *bob*, respectively assigned to roles *kids* and *parents*. The devices include *TV*, *DVD*, and *Playstation* which are assigned to the device role *Entertainment_Devices*. *EC* includes *weekends*, *evenings* and *TRUE*, respectively active on weekends, evening and always. The environment role *Entertainment_Time* is active when both environment conditions *weekends* and *evenings* are active while *Any_Time* is always active, as indicated in *EA*. *RPDRA* indicates that *kids* can access *Entertainment_Devices* when *Entertainment_Time* is active whereas *parents* are able to do so when *Any_Time* is active (i.e. always).

Table 3: EGRBAC Model Formalization

Users, Roles and Devices

$-U, R, D, P, OP, DR$ and *Constraints* are sets of users, roles, devices, permissions, operations, device roles, and constraints respectively

$-UA \subseteq U \times R$, many to one users to role assignment (home owner specified)

$-OPA \subseteq OP \times D$, a many to many assignment between operations and its target device (manufacturer specified)

$-P \subseteq D \times OP$, every permission is a mapping between an operation and its device (manufacturer specified)

– For convenience for every $p = (d_i, op_j) \in P$, let $p.d = d_i$ and $p.op = op_j$

– $OPPA \subseteq OP \times P$, a one to many operation to permissions association, where: $OPPA = \{(op_i, p_j) \mid p_j.op = op_i \wedge p_j \in P\}$

– $DPA \subseteq D \times P$, a one to many device to permissions association, where: $DPA = \{(d_i, p_j) \mid p_j.d = d_i \wedge p_j \in P\}$

– $PDRA \subseteq P \times DR$, a many to many permissions to device roles assignment (home owner specified)

Environment Roles and Environment Conditions

$-ER$ and EC are sets of environment roles and environment conditions respectively

$-EA \subseteq 2^{EC} \times ER$, many to many subsets of environment conditions to environment roles assignment (home owner specified)

Role Pairs

$-RP \subseteq R \times 2^{ER}$, a set of role pairs specifying all permissible combinations of a user role and subsets of environment roles (home owner specified)

– For convenience for every $rp = (r_i, ER_j) \in RP$, let $rp.r = r_i$ and $rp.ER = ER_j$

$-RRPA \subseteq R \times RP$, one to many role to role pairs association, where $RRPA = \{(r_m, rp_n) \mid rp_n \in RP \wedge rp_n.r = r_m\}$

$-ERPA \subseteq ER \times RP$, many to many environment roles to role pairs association, where $ERPA = \{(er_m, rp_n) \mid rp_n \in RP \wedge er_m \in rp_n.ER\}$

Role Pair Assignment

$-RPDRA \subseteq RP \times DR$, many to many role pairs to device roles assignment (home owner specified)

Authorization Predicate

– For a user u_i to perform operation op_k on device d_j when the set of environment conditions EC_m is active:

$(op_k, d_j) \in OPA \wedge$

$(\exists\, r_x \in R, rp_y \in RP, dr_l \in DR)$, where:

$(u_i, r_x) \in UA \wedge rp_y.r = r_x \wedge (rp_y, dr_l) \in RPDRA \wedge$

$((op_k, d_j), dr_l) \in PDRA \wedge$

$rp_y.ER \subseteq \{er \in ER \mid (\exists EC'_m \subseteq EC_m)[(EC'_m, er) \in EA]\}$

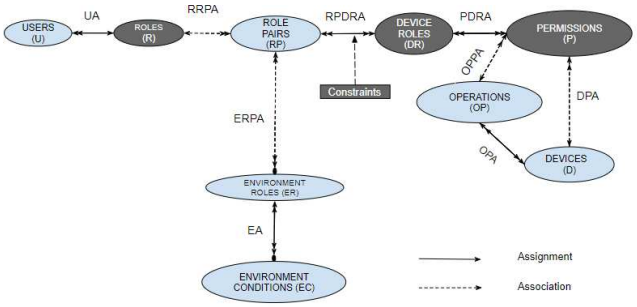

Figure 4: EGRBAC Model

7 EGRBAC Model for Smart Home IoT

In this section we define the EGRBAC (Extended Generalized Role-Based Access Control) model. EGRBAC is an extended version of GRBAC where authorization granularity is at the device-operation level rather than at the device level. Thus authorization can be given to, say, turn off an oven but not to turn it on. Furthermore, EGRBAC model eliminates the risk of policy conflicts as described in Section 8.1.2. EGRBAC model components, i.e. the sets, relations and functions, are shown in Figure 4, with the components which are different from those in GRBAC, or newly introduced in EGRBAC are shown in a different color. The formal definition given in Table 3, with the components and relations which are different from those in GRBAC, or newly introduced in EGRBA written in blue. In general, EGRBAC's components are similar to GRBAC's components, except for the differences discussed below.

First, a Role (R) here specifically represents the relationship between the user and the family. Unlike traditional RBAC systems, including GRBAC, EGRBAC requires each user to have a single role. In some rare cases you may have a user with two different roles, for example, a neighbour who is assigned to a neighbour role, but also happens to be a plumber who needs temporary access to repair the dishwasher, and so should have different set of privileges for that purpose. Assigning these additional privileges to the neighbour role will affect all neighbours, so is not appropriate. The best solution is to be able to give him another role say a plumber role, and assign those privileges to it. However, this will bring another issue of whether you want both roles to be active at the same time, and in that case do you want the user to get access to the permissions of the two roles at the same time. This issue can be solved by the concept of sessions. At this stage our goal is to build a formal, simple model which establishes a foundation for future more advanced access control models in smart homes, so we choose not to introduce the session concept for now. Second, here are two elements that are newly introduced in EGRBAC: Permissions (P), and Constraints. A permission is an approval to perform an operation on one device, in other words it is a mapping between an operation and its owner device. Each permission p has a device part $p.d$ which relates to the single device associated with it, and an operation part $p.op$ which relates to the single operation associated with it. Constraints are conditions that need to be satisfied when assigning device roles to role pairs, and will be discussed at the end of this section. Unlike in GRBAC where a device role is a way of categorizing devices, in EGRBAC a Device Role is a mean of categorizing permissions of different devices, e.g. in order to categorize the dangerous permissions of various smart devices, we can create a device role called dangerous devices and assign dangerous permissions (such as, turning on the oven, turning on the mower, and opening and closing the front door lock) to it. We could alternatively call this set Permission Categories but choose to retain the Device Roles name.

The main idea in EGRBAC as a whole is that a user is assigned to a single role and according to the current active environment roles some role pairs will be active, the user will get access to the permissions (not devices as in GRBAC) assigned to the device roles which are assigned to the current active role pairs. The arrangement of Table 3 is similar to Table 2, except for some modifications including UA to be many-to-one. In the top part which contains the users, roles, and devices related components and relations, we have a new set P which is specified as a subset of $D \times OP$, where each permission is a coupling between an operation and its target device. The manufacturer specifies the valid operations for each device. We also have two new associations, the first one is $OPPA$ a one-to-many association between operations and permissions, where we can have one operation for example Unlock that is assigned to the door lock device, and associated to three permissions with each permission on a different door lock. The second association is DPA a one-to-many association that map each permission to its target device. The DRA relation of GRBAC is replaced by $PDRA$, since in EGRBAC a device role categorizes permissions of different devices instead of categorizing devices.

The bottom part of Table 3 formalizes the authorization function of EGRBAC which is similar to the authorization function of GRBAC, except for item (d). Consider a user u_i who attempts to perform operation op_k on device d_j when the subset of environment conditions EC_m is active. This operation will succeed if and only if all of the following are true: (i) There is an OPA relation between op_k and d_j. (ii) There exists a role r_x in R, a role pair rp_y in RP, and a device role dr_l in DR, where: (a) User u_i is assigned role r_x in UA. (b) The role part of rp_y is r_x. (c) There is a $RPDRA$ relation between rp_y and dr_l. (d) There is a $PDRA$ relation between the permission (op_k, d_j) and dr_l. (e) The environment role part of rp_y is a subset of the current active environment roles that triggered by the current active environment conditions. The following use case scenarios illustrate the advantage of EGRBAC over GRBAC.

Use Case Scenario 1.B The objective is to allow kids during entertainment time to get access to a subset of capabilities in entertainment devices. For example, authorize G rated contents only on TV and DVD, and games rated 3, 7, and 12 on PlayStation, disallowing all else. Furthermore, authorize parents to enjoy all the permissions in entertainment devices at anytime. GRBAC is not capable enough to configure these policies since it has device granularity level. EGRBAC could be configured as shown in Figure 5 to achieve this objective. U, R, UA, and D are configured similar to Use case 1.A. We have the new component P, whereby each device has different permissions. We have two device roles: $Entertainment\_Devices$, and $Kids\_Friendly\_Content$, where we should assign all the permissions of TV, DVD, and $PlayStation$ to $Entertainment\_Devices$, and a subset of the permissions of these devices to $Kids\_Friendly\_Content$. EC, ER, EA, and RP are configured

$$
\begin{aligned}
&U = \{alex, bob, \ldots\},\ R = \{kids, parents, \ldots\}\\
&UA = \{(alex, kids), (bob, parents), \ldots\}\\
&D = \{TV, DVD, Playstation, \ldots\}\\
&P_{TV} = \{TV_G, TV_{PG}, TV_{PG13}, TV_R, TV_{NC-17}, \ldots\}\\
&P_{DVD} = \{DVD_G, DVD_{PG}, DVD_{PG13}, DVD_R, DVD_{NC-17}, \ldots\}\\
&P_{Playstation} = \{PS_{A3}, PS_{A7}, PS_{PG12}, PS_{A16}, PS_{A18}, PS_{BuyGames},\\
&\qquad PS_{InternetBrowsing}, PS_{Texting}, PS_{VoiceMessaging}, \ldots\}\\
&P = P_{TV} \cup P_{DVD} \cup P_{Playstation} \cup \ldots\\
&DR = \{Entertainment\_Devices, Kids\_Friendly\_Contents, \ldots\}\\
&PDRA = (P_{TV} \times \{Entertainment\_Devices\}) \cup\\
&\qquad (P_{DVD} \times \{Entertainment\_Devices\}) \cup\\
&\qquad (P_{Playstation} \times \{Entertainment\_Devices\}) \cup\\
&\qquad \{(TV_G, Kids\_Friendly\_Content),\\
&\qquad (DVD_G, Kids\_Friendly\_Contents)\} \cup\\
&\qquad (\{PS_{A3}, PS_{A7}, PS_{PG12}\} \times\\
&\qquad\qquad \{Kids\_Friendly\_Contents\}) \cup \ldots\\
&\\
&EC = \{weekends, evenings, TRUE \ldots\}\\
&ER = \{Entertainment\_Time, Any\_Time, \ldots\}\\
&EA = \{(\{weekends, evenings\}, Entertainment\_Time),\\
&\qquad (TRUE, Any\_Time), \ldots\}\\
&\\
&RP = \{(kids, \{Entertainment\_Time\}), (parents, \{Any\_Time\}), \ldots\}\\
&RPDRA = \{((kids, \{Entertainment\_Time\}),\\
&\qquad\qquad Kids\_Friendly\_Contents),\\
&\qquad ((parents, \{Any\_Time\}),\\
&\qquad\qquad Entertainment\_Devices), \ldots\}
\end{aligned}
$$

Figure 5: Use Case 1.B Configuration in EGRBAC

$$
\begin{aligned}
&U = \{alex, bob, \ldots\},\ R = \{kids, parents, \ldots\}\\
&UA = \{(alex, kids), (bob, parents), \ldots\}\\
&D = \{DoorLock, Oven, LawnMawer, \ldots\}\\
&P_{DoorLock} = \{DoorLock_{Lock}, DoorLock_{Unlock}, \ldots\}\\
&P_{Oven} = \{Oven_{On}, Oven_{Off}, \ldots\}\\
&P_{LawnMawer} = \{LawnMawer_{On}, LawnMawer_{Off}, \ldots\}\\
&P = P_{DoorLock} \cup P_{Oven} \cup P_{LawnMawer} \cup \ldots\\
&DR = \{Dangerous\_Devices, \ldots\}\\
&PDRA = (\{DoorLock_{Lock}, DoorLock_{Unlock}\} \times\\
&\qquad\qquad \{Dangerous\_Devices\}) \cup\\
&\qquad (\{Oven_{On}, Oven_{Off}\} \times\\
&\qquad\qquad \{Dangerous\_Devices\}) \cup\\
&\qquad (\{LawnMawer_{On}, LawnMawer_{Off}\} \times\\
&\qquad\qquad \{Dangerous\_Devices\}) \cup \ldots\\
&\\
&EC = \{TRUE, \ldots\}\\
&ER = \{Any\_Time, \ldots\}\\
&EA = \{(TRUE, Any\_Time), \ldots\}\\
&\\
&RP = \{(parents, \{Any\_Time\}), \ldots\}\\
&RPDRA = \{((parents, \{Any\_Time\}),\\
&\qquad\qquad Dangerous\_Devices), \ldots\}
\end{aligned}
$$

Figure 6: Use Case 2 Configuration in EGRBAC

$$
\begin{aligned}
&P_{barred} = \{DoorLock_{Lock}, DoorLock_{Unlock},\\
&\qquad Oven_{On}, Oven_{Off},\\
&\qquad LawnMawer_{On}, LawnMawer_{Off}\}\\
&R_{barred} = R - \{parents\}\\
&Constraints = \{(P_{barred}, R_{barred})\}
\end{aligned}
$$

Figure 7: Constraint Added to Use Case 2 Configuration in EGRBAC

similarly to Use case 1.A. *RPDRA* has two entries. The assignment of $(kids, \{Entertainment\_Time\})$ to device role $Kids\_Friendly\_Content$, this assignment specifies that users of role *kids* can use only the permissions assigned to $Kids\_Friendly\_Content$ when the environment role $Entertainment\_Time$ is active. The second assignment of $(parents, \{Any\_Time\})$ to $Entertainment\_Devices$ specifies that the users of role *parents* can use all the permissions assigned to $Entertainment\_Devices$ when the environment role Any_Time is active.

Use Case Scenario 2 The objective is to authorize only parents to use dangerous capabilities of dangerous devices (i.e lock, and unlock the smart door lock, switch on and off the smart oven and smart lawn mower) at any time. EGRBAC could be configured as shown in Figure 6 to achieve this.

Constraints in EGRBAC

An important component in EGRBAC is Constraints, whose consideration has been deferred so far. A constraint is an invariant that must be maintained at all times. Constraints are an integral part of RBAC and ABAC models [10, 21, 51]. In use case 2 the permissions embodied in the *Dangerous_Devices* role are assigned to the $(parents, \{Any\_Time\})$ role pair in *RPDRA*. However, this does not prevent future assignment of *Dangerous_Devices* to other role pairs, perhaps even to $(kids, \{Any\_Time\})$. This could happen inadvertently by the home owner. To prevent such situations, EGRBAC incorporates constraints that forbid assigning specific permissions to specific roles. Formally, $Constraints \subseteq 2^P \times 2^R$ constitute a many to many subset of permissions to subset of roles relation. Each $c = (P_i, R_j) \in Constraints$ specifies the following invariant for every $p_m \in P_i$ and every $r_n \in R_j$:

$$\forall_{rp_p \in RP, dr_q \in DR}(rp_p, dr_q) \in RPDRA \implies (p_m, dr_q) \notin PDRA \vee (rp_p.r \neq r_n)$$

Thus, it is forbidden to assign any device role that p_m is assigned to, to any role pair that r_n is the role part of it. Use case 2 can be augmented with the constraint shown in Figure 7 to ensure that the permissions enumerated in P_{barred} cannot be assigned beyond the *parents* role.

8 EGRBAC Analysis and Limitations

8.1 Model Analysis

8.1.1 Our Developed Desirable Characteristics As described in Section 5, our threat model focusses on insiders, i.e. those who have legitimate digital and physical access to the house. External adversaries or legitimate users who try to exploit system vulnerabilities to carry out attacks are outside the scope of our work.

Some criteria for a suitable access control model were discussed in Section 2. While these criteria may not be complete, they are arguably desirable for any such model. In the following we analyze our model based on those specifications.

Dynamic We consider our model as a dynamic model. Environment conditions and environment roles allow us to give different users access rights to different capabilities under specific environment contextual factors. Moreover, device roles enable our model to give users access to some permissions, or to some devices based on different contextual characteristics.

Fine grained As illustrated in Use case 1.B, and Use case 2, our model is able to give users access to some permissions within a single device without the need to give them the access to the entire device, which makes it a capability centric model instead of a device centric model.

Suitable for constrained home environment Our model is suitable for constrained smart home environment for two main reasons: (a) It is built on top of RBAC model, which is considered

as a simple model. EGRBAC exploits the organizational power of roles for grouping environment states and objects, in addition to subjects. It authorizes or revokes access based on roles instead of on an individual basis. Establishing a set of roles in a small or medium-sized environment such as houses is not a challenging task. (b) The enforcement architecture that we adopt (see Section 4) includes the component smart hub, which facilitates transferring the policy decision engine to a third local device. This enables devices to collect and analyze data externally, but closer to the source of information, react autonomously to local events, and communicate securely with each other on local networks. Having such setting allows smart homes to enforce EGRBAC model without the need to incorporate advanced or computationally heavy smart things. Moreover, mediating each request through smart hub instead of directly accessing the smart devices solves the heterogeneity problem of IoT devices.

Designed and interpreted for smart home IoT Our model is developed to fit smart home IoT access control challenges. This model is dynamic, fine grained , and captures the complex relationships between home users.

Implemented and tested Our model is demonstrated with three illustrative use cases, an AWS implementation that captures local, and remote access for smart home devices as described in Section 9, and a performance analysis.

U-D or D-D Our model ia a U-D access control model.

Provides a formal Access Control Model Our model is formally structured, and defined as illustrated in Section 7.

8.1.2 Policy Conflicts Conflicting policies may occur when you have negative policies, where you prevent specific roles from accessing specific permissions. In EGRBAC model our policies are positive policies where you give roles access to specific permissions by assigning appropriate role pairs to appropriate device roles. Instead of negative policies, EGRBAC uses constraints to prevent a specific role r_n from accessing a specific permission p_m (see Section 7).

8.1.3 Usability and Expressiveness One of the important aspects that need to be considered in smart home access control models is usability, since smart home residents are usually constrained, and not willing to deal with complicated systems. We believe that at this point it would be premature to conduct a usability study for EGRBAC. This model is our first step toward building a set of models ranging from relatively simple and complete to incorporating increasingly sophisticated and comprehensive features. It still needs to be further developed and extended. Another important aspect is expressiveness, wether the system is capable of expressing policies that depict users requirements. Our smart home IoT access control model criteria which is introduced in Section 2 incorporates the new perspective of access control specifications recently introduced by He et al [26] as explained in Section 2. He et al validated the expressiveness of their specifications by conducting an online survey-based user study of 425 participants. As we discussed earlier in Section 8 our model meets our criteria which implies that it meets He et al perspective. However, in order to deploy this model for commercial uses, a more general sophisticated expressiveness study should be conducted.

8.2 Limitations

Our model can be improved upon in several ways. Except for relationships, it doesn't capture other user attributes. Moreover, it does not handle device to device communication. Furthermore, our model restrict each user to one role only. Finally, it doesn't consider continuous verification for access control authorized policies, where the authorization predicate is only examined at the time of request but does not support ongoing controls for relatively long-lived operations or for immediate revocation.

9 Proof-Of-Concept Implementation

In this section we describe a proof-of-concept implementation of EGRBAC. We demonstrate a consolidated use case as shown in Figure 8. This incorporates two of our earlier use cases, viz., 1.A and 2 with some additions. Here we have five users, *alex*, *bob*, *susan*, *james*, and *julia*, five roles, *kids*, *parents*, *babySitters*, *guests*, and *neighbors*, and five devices *DoorLock*, *Oven*, *TV*, *DVD*, and *Playstation*. Each device has two operations, lock/unlock for *DoorLock*, and on/off for the remaining devices. Moreover, we have two device roles *Dangerous_Devices*, and *Entertainment_Devices*. The lock/unlock permissions on *DoorLock*, and on/off permissions on *Oven* are assigned to the device role *Dangerous_Devices*, while all permissions of *TV*, *DVD*, and *Playstation* are assigned to the device role *Entertainment_Devices*. The *EC* includes *weekends*, *evenings* and *TRUE*, respectively active on weekends, evening and always. The environment role *Entertainment_Time* is active when both environment conditions *weekends* and *evenings* are active while *Any_Time* is always active, as indicated in *EA*. *RPDRA* indicates that *parents* can access *Dangerous_Devices*, and *Entertainment_Devices* when *Any_Time* is active (i.e. always), *kids* can access *Entertainment_Devices* when *Entertainment_Time* is active, *babySitters*, *guests*, and *neighbors* can access *Entertainment_Devices* when *Any_Time* is active.

We simulated the consolidated use case using AWS IoT service [1]. The simulation illustrates how the access control model and policies can be configured to establish the applicability of our model utilizing commercially available systems. An AWS account is required to configure and deploy the AWS IoT service known as Greengrass. The Greengrass SDK extends cloud capabilities to the edge, which in our case is the smart home. This enables devices to collect and analyze data closer to the source of information, react autonomously to local events, and communicate securely with each other on local networks[2]. In our system Greengrass serves as a smart hub and a policy engine. It runs on a dedicated virtual machine. Through AWS IoT management console, one virtual object (aka digital shadow) is created for each physical device and the two are cryptographically linked via digital certificates with attached authorization policies. The physical devices that we simulated in our use cases are: smart door lock, smart oven, smart TV, smart DVD, and smart Playstation. Each simulated device is run on a separate virtual machine. These devices use MQTT protocol to communicate to the AWS IoT service with TLS security. Since the environment conditions in our use case are time based, they are directly sensed by Greengrass. To enforce EGRBAC, we utilized two Json files UserRoleAssignment.json and policy.json, where UserRoleAssignment.json defines the assignments of users to their corresponding roles while policy.json defines all other EGRBAC components relevant to the use case. We also utilized the

$$
\begin{aligned}
&U = \{alex, bob, susan, james, julia\}\\
&R = \{kids, parents, babySitters, guests, neighbors\}\\
&UA = \{(alex, kids), (bob, parents), (susan, babySitters),\\
&\qquad (james, guests), (julia, neighbors)\}\\
&D = \{DoorLock, Oven, TV, DVD, Playstation\}\\
&P_{DoorLock} = \{DoorLock_{Lock}, DoorLock_{Unlock}\}\\
&P_{Oven} = \{Oven_{On}, Oven_{Off}\}\\
&P_{TV} = \{TV_{On}, TV_{Off}\}\\
&P_{DVD} = \{DVD_{On}, DVD_{Off}\}\\
&P_{Playstation} = \{PS_{On}, PS_{Off}\}\\
&P = P_{DoorLock} \cup P_{Oven} \cup P_{TV} \cup P_{DVD} \cup P_{Playstation}\\
&DR = \{Dangerous\_Devices, Entertainment\_Devices\}\\
&PDRA = (\{DoorLock_{Lock}, DoorLock_{Unlock}\} \times\\
&\qquad \{Dangerous\_Devices\}) \cup\\
&\qquad (\{Oven_{On}, Oven_{Off}\} \times\\
&\qquad \{Dangerous\_Devices\}) \cup\\
&\qquad (P_{TV} \times \{Entertainment\_Devices\}) \cup\\
&\qquad (P_{DVD} \times \{Entertainment\_Devices\}) \cup\\
&\qquad (P_{Playstation} \times \{Entertainment\_Devices\})\\
&\\
&EC = \{weekends, evenings, TRUE\}\\
&ER = \{Entertainment\_Time, Any\_Time\}\\
&EA = \{(\{weekends, evenings\}, Entertainment\_Time),\\
&\qquad (TRUE, Any\_Time)\}\\
&\\
&RP = \{(kids, \{Entertainment\_Time\}), (parents, \{Any\_Time\}),\\
&\qquad (babySitters, \{Any\_Time\}), (guests, \{Any\_Time\}),\\
&\qquad (neighbors, \{Any\_Time\})\}\\
&\\
&RPDRA = \{((parents, \{Any\_Time\}), Dangerous\_Devices),\\
&\qquad ((kids, \{Entertainment\_Time\}), Entertainment\_Devices),\\
&\qquad ((parents, \{Any\_Time\}), Entertainment\_Devices),\\
&\qquad ((babySitters, \{Any\_Time\}), Entertainment\_Devices),\\
&\qquad ((guests, \{Any\_Time\}), Entertainment\_Devices),\\
&\qquad ((neighbors, \{Any\_Time\}), Entertainment\_Devices)\}
\end{aligned}
$$

Figure 8: Proof-of-Concept Use Case

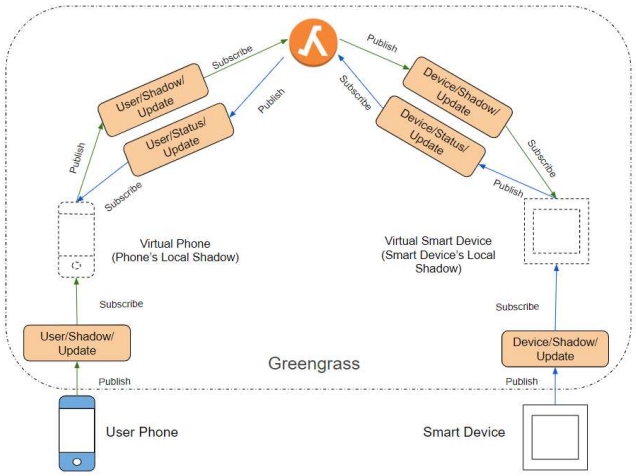

Figure 9: Local Request Processing

lambda function service in AWS IoT platform to receive the operation requests of users to access the smart devices in the house, analyze each request according to the content of the policy.json and UserRoleAssignment.json files, and finally trigger the desired actions on the corresponding simulated devices. The lambda function, the UserRoleAssignment.json file, and the policy.json file are all configured in the Greengrass. The following use cases scenarios provide additional implementation details. We should mention that our system is a default deny system.

Figure 9, illustrates how the communication is handled in our implementation when the user tries to send operation request to turn on a smart TV through his mobile phone while he is inside the

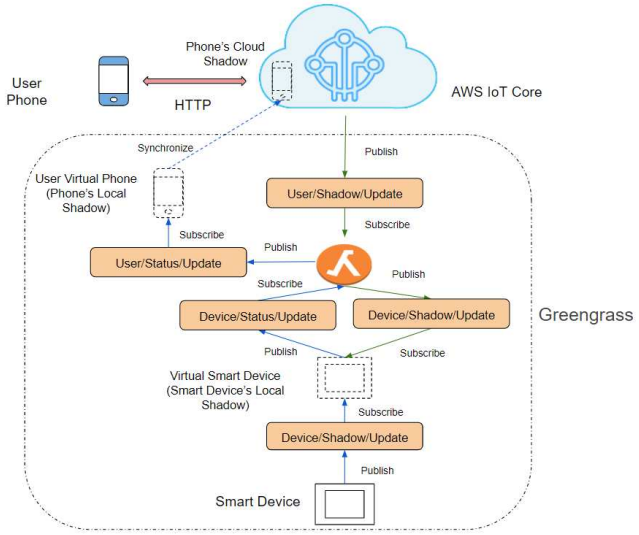

Figure 10: Remote Request Handling in Our System

house. In this case, a request is sent via MQTT protocol to the virtual object (or local shadow) corresponding to his phone in Greengrass. There is a publish/subscribe relation between the user phone, and the local shadow through the user private topic *User/Shadow/Update*. The user phone will publish to the topic *User/Shadow/Update*, and the local shadow will get notified with the request. After that the local shadow publishes to the user private topic *User/Shadow/Update*, and then since the lambda function is subscribed to this topic it analyzes the request according to the policy.json and UserRoleAssignment.json files and makes a decision wether to allow the user to turn on the TV or not. At this point, there are two cases, either permission is granted or denied. If permission is denied, the lambda function publishes to the user public topic *User/Status/Update*, the local shadow gets notified and updates the user phone that the permission was denied. The smart TV in this case does not get an indication that a user attempted to access it. If permission is granted, the smart TV local shadow is notified through the device private topic *Device/Shadow/Update* and updates the smart TV with the turn on command. After the smart TV is turned on, it publishes to the device private topic *Device/Shadow/Update* and the TV local shadow is notified which further notifies the lambda function by publishing to the device public topic *Device/Status/Update*. The lambda function then notifies the user phone local shadow by publishing to the topic *User/Status/Update*, and finally, the user phone local shadow updates the user phone that the TV was turned on successfully. Figure10, illustrates how the communication is handled in our implementation in case of remote access. Let us assume that the user Bob is trying to turn on the oven using his smart phone from a remote place (the user is outside the house, hence his phone cannot directly access the green grass). First, a request is sent through the HTTP send protocol to the cloud's synchronized shadow state of the user device, in this case the users phone. Once the users phone state is changed on the cloud, the cloud forwards the message to the local Greengrass lambda by publishing to the user private topic

Table 4: One User Sending Requests to Multiple Devices

Number of Users	Number of devices	Lambda Processing Time in ms.	Total Number of requests
1	1	1.029138	1000
1	3	1.236029	3000 (1000 per request)
1	5	1.202856	5000 (1000 per request)

Table 5: One User Sending Requests to One Device

Number of Users	Number of devices	Lambda Processing Time in ms.	Total Number of requests
1	1	1.029138	1000
3	3	1.796938	3000 (1000 per request)
5	5	2.833097	5000 (1000 per request)

Table 6: Multiple Users Sending Requests to One Device

Number of Users	Number of devices	Lambda Processing Time in ms.	Total Number of requests
1	1	1.029138	1000
3	1	0.955529	3000 (1000 per request)
5	1	0.956221	5000 (1000 per request)

User/Shadow/Update, the lambda receives the request, analyzes it according to the access control policies defined in the policy.json file and the user assignments relations defined in the UserRoleAssignment.json file and makes a decision wether to allow the user to turn on the oven or no. If the decision was to allow the user to turn on the oven, the lambda function would send the request to the smart device Greengrass's local shadow by publishing to the device's private topic *Device/Shadow/Update*, the local shadow would get the request and automatically update the smart device (smart oven in this case) to turn on. When the smart device perform the operation, it notifies its local shadow by publishing to the device private topic *Device/Shadow/Update*, the local shadow then notifies the lambda by publishing to the device public topic *Device/Status/Update*, the lambda then updates the user phone local shadow by publishing to the user public topic *User/Status/Update*, when the user phone local shadow get notified it automatically synchronize this state to the cloud shadow which in turn notifies the user phone that the request has been served. On the other hand, if the decision was not to allow this operation to be performed, the lambda function would publish to the user public topic *User/Status/Update*, the local shadow would get notified and would automatically synchronize this sate to the cloud's synchronized shadow state of the device. The cloud's shadow would then update the user phone through the http send protocol that the permission was denied and the user has no right to turn on the oven. The smart oven, in this case, would never get an indication that a user attempted to access it.

9.1 Performance results

We executed multiple test cases to measure the processing time in different scenarios. In our performance testing, we implemented the configuration of Figure 8. In the following test cases, we measure the average lambda function execution time under different conditions. Table 4 shows the average lambda function execution time when we send multiple requests from one user to multiple devices. The first row shows the average time when the parent Bob requests to unlock the door lock. The second row shows the average time when Bob requests to turn on the oven, the TV, and the DVD at the same time. The third row shows the average time when Bob requests to unlock the door lock, turn on the oven, the TV, the DVD, and the playStation at the same time. All the requests were approved as they were supposed to according to our configured policies. We can notice that for the same user the lambda processing time increases when the user increases the number of requests sent to different devices at the same time. Table 5 shows the average lambda function execution time when we send multiple requests from multiple users to multiple devices (one user per device) at the same time. The first row shows the average time when the parent Bob requests to unlock the door lock. The second row shows the average time when Bob requests to unlock the door lock, the kid Alex requests to turn on the oven, and the babysitter Susan requests to turn on the TV at the same time. The third row shows the average time when the three access requests tested in the second row are carried again in addition to, the guest James requests to turn on the DVD, and the neighbor Julia requests to turn on the playStation. The system responded correctly where all the requests were approved except for when the kid Alex requests to turn on the oven. We can conclude that when the number of requests for different users and different devices (one user per device) increases, the lambda processing time also increases. Finally, Table 6 shows the average lambda function execution time when we send multiple requests from multiple users to one device at the same time. The first, second, and third rows show the average time when the parent (1 user), the parent the kid and the babysitter (3 users), or the parent the kid the babysitter the guest and the neighbor (5 users) respectively all request to unlock the door at the same time. The system responded correctly where all the requests were denied except for when the parent Bob requests to unlock the door lock. Here, we can see that the average of the lambda processing time decrease when we have more denies. This result is expected since our policy checking engine (the lambda function) implemented to check for the authorization predicate explained in Table 3. In order to approve a request the lambda function need to verify each condition in the authorization predicate. On the other hand, if only one of the authorization predicate conditions is violated the lambda function will deny the request without the need to check the rest of the authorization predicate. To conclude, our system takes more time when approving a request than when denying it.

Overall, our model is functional, and can be easily applied. Moreover, we can notice that the execution time is generally low. However, it is directly proportional with the number of requests if the requests are approved, and indirectly proportional with the number of requests if the requests are denied.

10 Conclusion and Future Directions

In this paper, we provide our formalization for GRBAC model previously proposed by [14], and build upon it to propose the EGRBAC access control model for smart home IoT. Our model's main goal is to insure that legitimate users are only permitted to use the devices which they are allowed to access under the appropriate conditions. It fills the gap in the area of access control model for smart home IoT. It is a dynamic, fine-grained model that grants access based on the specific permission required rather than at device granularity. Our model meets the new perspective of smart home IoT access control requirements recently identified by [26]. We demonstrated

our model with three different use cases. The model includes one type of constraint to prevent poor or insecure policy configurations.

Furthermore, we analyze IoT access control models proposed in the literature based on the criteria mentioned in Table 1, and compared them to our model. This criteria was developed in response to many studies that have been conducted to identify IoT access control security and privacy vulnerabilities as discussed in Section 2. In Section 8 we analyze our model according to the criteria introduced in Table 1. Moreover, in Section 9, we illustrate our model with a proof of concept implementation in AWS, we also conducted a performance test to depict how our system responds in different scenarios with different loads, the results show that our model is functional, and applicable. In Section 4 we provide an overview of our adopted home IoT enforcement architecture. Our model still need some further work as discussed in Section 8.2. In the future we are planning, to develop a family (or series) of models ranging from relatively simple and complete to incorporating increasingly sophisticated and comprehensive features.

References

[1] [n.d.]. AWS-IoT. https://aws.amazon.com/iot/. Online; Accessed: 2019-11-07.
[2] [n.d.]. AWS IoT Greengrass. https://docs.aws.amazon.com/greengrass/latest/developerguide/what-is-gg.html. Online; Accessed: 2019-11-07.
[3] T. Ahmed, et al. 2017. Classifying and comparing attribute-based and relationship-based access control. In *CODASPY '17*. ACM.
[4] M. Alramadhan and K. Sha. 2017. An overview of access control mechanisms for internet of things. In *ICCCN*. IEEE.
[5] O. Arias, et al. 2015. Privacy and security in internet of things and wearable devices. *TMSCS* (2015).
[6] S. Bandara, et al. 2016. Access control framework for api-enabled devices in smart buildings. In *APCC*. IEEE.
[7] E. Barka, et al. 2015. Securing the web of things with role-based access control. In *C2SI*. Springer.
[8] E. Barka and R. Sandhu. 2000. Framework for role-based delegation models. In *ACSAC*. IEEE.
[9] B. Bezawada, et al. 2018. Securing Home IoT Environments with Attribute-Based Access Control. In *ABAC'18*. ACM.
[10] K. Z. Bijon, et al. 2013. Towards an Attribute Based Constraints Specification Language. In *SOCIALCOM '13*. https://doi.org/10.1109/SocialCom.2013.23
[11] E. Carniani, et al. 2016. Usage control on cloud systems. *Future Gen. Comp. Sys.* (2016).
[12] Y. Cheng, et al. 2012. Relationship-based access control for online social networks: Beyond user-to-user relationships. In *SocialCom*. IEEE.
[13] M. J. Covington, et al. 2001. Securing context-aware applications using environment roles. In *SACMAT '01*. ACM.
[14] M. J. Covington, et al. 2000. *Generalized role-based access control for securing future applications*. Technical Report. Georgia Tech.
[15] A. Cui and S. J. Stolfo. 2010. A quantitative analysis of the insecurity of embedded network devices: results of a wide-area scan. In *ACSAC '10*. ACM.
[16] L. M. S. De Souza, et al. 2008. Socrades: A web service based shop floor integration infrastructure. In *The IoT*. Springer.
[17] T. Denning, et al. 2013. Computer security and the modern home. *Commun. ACM* (2013).
[18] A. Dorri, et al. 2017. Blockchain for IoT security and privacy: The case study of a smart home. In *PerCom workshops*. IEEE.
[19] E. Fernandes, et al. 2016. Security analysis of emerging smart home applications. In *SP*. IEEE.
[20] E. Fernandes, et al. 2016. Flowfence: Practical data protection for emerging iot application frameworks. In *USENIX Security 16*.
[21] D. F. Ferraiolo, et al. 2001. Proposed NIST standard for role-based access control. *TISSEC* (2001).
[22] P. W. Fong, et al. 2009. A privacy preservation model for facebook-style social network systems. In *ESORICS*. Springer.
[23] D. Geneiatakis, et al. 2017. Security and privacy issues for an IoT based smart home. In *2017 40th MIPRO*. IEEE.
[24] J. Granjal, et al. 2015. Security for the internet of things: a survey of existing protocols and open research issues. *IEEE Comm. Surv. & Tutorials* (2015).
[25] Z. Guoping and G. Wentao. 2011. The research of access control based on UCON in the internet of things. *Journal of Software* (2011).
[26] W. He, et al. 2018. Rethinking access control and authentication for the home internet of things (IoT). In *USENIX Security 18*.
[27] K. Hill. 2013. Baby Monitor Hack Could Happen To 40,000 Other Foscam Users. https://www.forbes.com/sites/kashmirhill/2013/08/27/baby-monitor-hack-could-happen-to-40000-other-foscam-users/613ec55458b5.
[28] G. Ho, et al. 2016. Smart locks: Lessons for securing commodity internet of things devices. In *ASIA CCS '16*. ACM.
[29] V. C. Hu, et al. 2015. Attribute-based access control. *Comp.* (2015).
[30] E. V. H. J. Dennis. 1966. Programming semantics for multiprogrammed computations. In *Comm. of the ACM*.
[31] X. Jin, et al. 2012. A unified attribute-based access control model covering DAC, MAC and RBAC. In *IFIP Annual Conf. on Data and App. Sec.* Springer.
[32] J. Jindou, et al. 2012. Access control method for web of things based on role and sns. In *CIT 2012*. IEEE.
[33] S. Kaiwen and Y. Lihua. 2014. Attribute-role-based hybrid access control in the internet of things. In *APWeb*. Springer.
[34] A. La Marra, et al. 2017. Implementing usage control in internet of things: A smart home use case. In *2017 IEEE Trustcom/BigDataSE/ICESS*. IEEE.
[35] A. Lazouski, et al. 2012. Usage control in cloud systems. In *ICITST*. IEEE.
[36] I. Lee and K. Lee. 2015. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons* 58, 4 (2015).
[37] J. Liu, et al. 2012. Authentication and access control in the internet of things. In *2012 32nd Int. Con. on Dist. Comp. Sys. Workshops*. IEEE.
[38] F. Martinelli, et al. 2018. Too long, did not enforce: a qualitative hierarchical risk-aware data usage control model for complex policies in distributed environments. In *CPSS '18*. ACM.
[39] T. Matthews, et al. 2017. Stories from survivors: Privacy & security practices when coping with intimate partner abuse. In *CHI'17*. ACM.
[40] P. Morgner, et al. 2016. All your bulbs are belong to us: Investigating the current state of security in connected lighting systems. *CoRR* (2016).
[41] A. Mutsvangwa, et al. 2016. Secured access control architecture consideration for smart grids. In *IEEE PES PowerAfrica*. IEEE.
[42] S. Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. \T1\ textquotedblrighthttp://bitcoin.org/bitcoin.pdf.
[43] O. Novo. 2018. Blockchain meets IoT: An architecture for scalable access management in IoT. *IEEE IoT Journal* (2018).
[44] T. Oluwafemi, et al. 2013. Experimental security analyses of non-networked compact fluorescent lamps: A case study of home automation security. In *LASER 2013*.
[45] A. Ouaddah, et al. 2017. Towards a novel privacy-preserving access control model based on blockchain technology in IoT. In *Europe and MENA Coop. Adv. in Inf. and Comm. Tech.* Springer.
[46] A. Ouaddah, et al. 2017. Access control in the Internet of Things: Big challenges and new opportunities. *Comp. NW* 112 (2017).
[47] J. Park. 2003. *Usage control: A unified framework for next generation access control*. Ph.D. Dissertation. George Mason University.
[48] J. Park and R. Sandhu. 2002. Towards usage control models: beyond traditional access control. In *SACMAT '02*. ACM.
[49] E. Rissanen. 2013. eXtensible Access Control Markup Language (XACML) Version 3.0. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html.
[50] R. S. Sandhu. 1998. Role-based access control. In *Advances in computers*. Vol. 46. Elsevier.
[51] R. S. Sandhu, et al. 1996. Role-based access control models. *Comp.* (1996).
[52] S. Sicari, et al. 2015. Security, privacy and trust in Internet of Things: The road ahead. *Computer networks* (2015).
[53] P. Spiess, et al. 2009. SOA-based Integration of the Internet of Things in Enterprise Services. In *ICWS*. IEEE Comp. Soc. Press.
[54] A. Tilley. 2016. How A Few Words to Apple's Siri Unlocked a Man's Front Door. http://www.forbes.com/sites/aarontilley/2016/09/21/apple-homekit-siri-security.
[55] B. Ur, et al. 2013. The current state of access control for smart devices in homes. In *HUPS*.
[56] G. Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014).
[57] Y. Xie, et al. 2015. Three-layers secure access control for cloud-based smart grids. In *IEEE 82nd VTC2015-Fall*. IEEE.
[58] N. Ye, et al. 2014. An efficient authentication and access control scheme for perception layer of internet of things. *Applied Math. & Inf. Sciences* (2014).
[59] G. Zhang and J. Tian. 2010. An extended role based access control model for the Internet of Things. In *2010 ICINA*. IEEE.
[60] X. Zhang, et al. 2005. Formal model and policy specification of usage control. *TISSEC* (2005).
[61] Y. Zhang and X. Wu. 2016. Access control in internet of things: A survey. *arXiv preprint arXiv:1610.01065* (2016).