Hybrid Approaches (ABAC and RBAC) Toward Secure Access Control in Smart Home IoT

Safwa Ameer, James Benson, and Ravi Sandhu, *Fellow, IEEE*

Abstract—Smart homes are interconnected homes in which a wide variety of digital devices with limited resources communicate with multiple users and among themselves using multiple protocols. The deployment of resource-limited devices and the use of a wide range of technologies expand the attack surface and position the smart home as a target for many potential security threats. Access control is among the top security challenges in smart home IoT. Several access control models have been developed or adapted for IoT in general, with a few specifically designed for the smart home IoT domain. Most of these models are built on the role-based access control (RBAC) model or the attribute-based access control (ABAC) model. However, recently some researchers demonstrated that the need arises for a hybrid model combining ABAC and RBAC, thereby incorporating the benefits of both models to better meet IoT access control challenges in general and smart homes requirements in particular. In this paper, we used two approaches to develop two different hybrid models for smart home IoT. We followed a role-centric approach and an attribute-centric approach to develop HyBAC$_{RC}$ and HyBAC$_{AC}$, respectively. We formally define these models and illustrate their features through a use case scenario demonstration. We further provide a proof-of-concept implementation for each model in Amazon Web Services (AWS) IoT platform. Finally, we conduct a theoretical comparison between the two models proposed in this paper in addition to the EGRBAC model (RBAC model for smart home IoT) and HABAC model (ABAC model for smart home IoT), which were previously developed to meet smart homes' challenges.

Index Terms—IoT, smart homes, access control, ABAC, RBAC

1 Introduction

Currently, the Internet of Things (IoT) is a key topic in technology. One of the most popular domains for deploying smart connected devices is the smart home. The smart home consists of a network of physical objects (things) equipped with sensors, software, and other technologies that enable it to exchange data and information with other devices, users, and systems over the Internet. Smart homes' main purposes are to anticipate and respond to the needs of the occupants, working to promote their comfort, convenience, security, and entertainment through the management of technology within the home and connections to the world beyond [1]. Several real-world examples have shown the shortcomings of current access control policy specification and authentication for home IoT devices, as described in [2], [3], and [4]. As illustrated in [2], the characteristics that make IoT distinct from prior computing domains necessitate a rethinking of access control and authentication. In the literature, several access control models have been proposed for IoT in general, with a few specifically designed to meet smart homes' challenges. Most models are based on ABAC or RBAC. RBAC has been argued to be more suitable for IoT due to its simplicity in management and review, whereas ABAC management and review tasks are more complex [3, 26, 27]. Furthermore, RBAC enforcement may also be more lightweight for constrained home smart devices. Nevertheless, there are those who argue that ABAC models are more scalable and dynamic due to the fact that they can capture contextual information specific to different devices and environmental conditions [5], [6], [7].

Recently, some researchers showed that while RBAC-based models are simpler in management and review, they are not capable enough to capture the entire dynamic characteristics of the IoT environment [8], [9], [10], [11], [12], [13]. On the other hand, in ABAC-based models, it can be very complicated to determine and limit the permissions available for each user at assignment time. This can make it infeasible to determine risk exposure for a given user [10], [14]. Hence, as several authors have suggested [8], [9], [10], [11], [12], [13], [14], in order to overcome these challenges, a hybrid model combining ABAC and RBAC characteristics is needed.

In this paper, we introduce two hybrid models, HyBAC$_{RC}$ and HyBAC$_{AC}$, that cover different authorizations for every possible user's, environment's, operation's, and device's static or dynamic condition while combining the advantages of ABAC and RBAC based models. We formally define each model and illustrate it with a use case scenario and proof of concept implementation. In developing these models, we started with EGRBAC (a role-based access control model) and HABAC (an attribute-based access control model). We then followed two different approaches: a role-centric approach in developing HyBAC$_{RC}$ and an attribute-centric approach in

• *The authors are with the Institute for Cyber Security (ICS), NSF Center for Security and Privacy-Enhanced Cloud Computing (C-SPECC), Department of Computer Science, University of Texas at San Antonio (UTSA), San Antonio, TX 78249 USA. E-mail: safwa.ameer@gmail.com, {james.benson, ravi.sandhu}@utsa.edu.*

Manuscript received 30 November 2021; revised 1 October 2022; accepted 8 October 2022. Date of publication 21 October 2022; date of current version 1 September 2023.
This work was supported in part by NSF CREST-PRF under Award 2112590 and in part by NSF CREST under Grant HRD1736209.
(Corresponding author: Safwa Ameer.)
This article has supplementary downloadable material available at https://doi.org/10.1109/TDSC.2022.3216297, provided by the authors.
Digital Object Identifier no. 10.1109/TDSC.2022.3216297

developing HyBAC$_{AC}$ [14]. The reason for choosing EGRBAC and HABAC as base models is that they are both contextual aware and fine-grained models explicitly designed to meet smart home challenges. HyBAC$_{RC}$ and HyBAC$_{AC}$ maintain the criteria for smart home IoT access control models proposed by [15], and the new perspective of smart home IoT access control requirements recently identified by [2]. Moreover, we conducted a comprehensive theoretical comparison between the two models proposed in this paper, viz. HyBAC$_{RC}$ and HyBAC$_{AC}$, and the two earlier models of EGRBAC and HABAC. The comparison is based on criteria adapted and developed from [16].

HyBAC$_{RC}$ and HyBAC$_{AC}$ are proposed for the smart home IoT environment. However, they can be adjusted and adapted for other IoT application domains. Our ultimate goal is to have a family of access control models ranging from relatively simple to more sophisticated with better features and more expressiveness power to provide policy designers with a range of models to choose from according to the environment requirements and the business needs.

The structure of this paper is as follows. Section 2 motivates the paper. Section 3 provides an analysis and review of related work. Section 4 introduces HyBAC$_{RC}$. It provides a formal definition for HyBAC$_{RC}$ along with an illustrative use case scenario. In Section 5, we introduce HyBAC$_{AC}$ followed by a formal definition and an illustrative use case. In Section 6, we demonstrate our models with a proof of concept implementation. In Section 7, we analyze and compare the two new hybrid models along with the earlier EGRBAC and HABAC. Moreover, in Section 8, we discuss the paper. Finally, Section 9 concludes the paper.

2 Motivation

Role-based access control (RBAC) and attribute-based access control (ABAC) models have been extensively studied in the literature. However, as discussed earlier, in light of the fact that, unlike RBAC, ABAC models can capture different devices and environment contextual information, some researchers believe they are more scalable, fine-grained, and dynamic. This would suggest that ABAC is more suitable for IoT access control [5], [6], [7]. Alternatively, other researchers argue that RBAC models are more suitable for IoT due to their simplicity in management and review, as well as their lightweight enforcement for constrained IoT devices [8], [17], [18]. Hence, as expressed in [10], [15], [19] it is not completely clear what are the advantages of ABAC over RBAC and vice versa when it comes to smart home IoT in specific and smart IoT application domains in general.

Recently Ameer et al. [15] introduced the EGRBAC model for smart home IoT [15], in which they extended an RBAC model to propose a dynamic model that can capture different permissions, devices, and environmental characteristics, thereby solving the RBAC limitation of capturing devices and environment information. Shortly after, they proposed the HABAC model for smart home IoT [10], an ABAC-based model specifically designed to meet smart home IoT requirements. Moreover, they performed a theoretical comparison between the expressiveness power of the two models, where they concluded that a hybrid model combining EGRBAC and HABAC characteristics would better capture access control requirements in the smart home IoT environment. A similar conclusion was also derived for other IoT application domains by different researchers in the literature [8], [9], [10], [11], [12], [13]. These researchers reiterated the need for hybrid models that combine ABAC and RBAC-based models' advantages while mitigating their disadvantages. In this research, we further motivate the need for hybrid access control models for smart home IoT that combine the advantages of the EGRBAC model (which is an extended RBAC-based model for smart home IoT) and the HABAC model (which is an extended ABAC-based model for smart home IoT) for the following reasons.

1. Role-based Access control models are incapable of handling dynamic attributes.

In general, to obtain a dynamic access control model, we have two types of attributes that need to be expressed and used in authorization policies: static attributes and dynamic attributes [14]. Static attributes have relatively fixed values over a long period. Setting and changing the values of static attributes typically requires administrator intervention, for instance, the user relationship to the house, user skill set, the danger level of the device operations, and device owner. On the other hand, dynamic attributes reflect contextual properties that can change at any time due to various circumstances, possibly rapidly and unpredictably, such as time of the day, user location, and device temperature. Values of dynamic attributes are automatically determined by sensors deployed in the smart home under homeowner control.

RBAC-based models, including EGRBAC, can effectively capture static user attributes and in some models, static device attributes and static and dynamic environment attributes. On the other hand, expressing users' and devices' dynamic attributes in RBAC models, including EGRBAC, creates significant difficulties and can be costly and cumbersome for two principal reasons. The multiplicity of combinations that need to be considered can lead to role explosion. Moreover, RBAC-based models, including EGRBAC, lack mechanisms to dynamically activate and deactivate different users, sessions, and roles according to varying dynamic characteristics. For example, consider a use case where the homeowner wants to permit teenagers to use the front door lock permissions in some exceptional circumstances when they are granted a token by one of their parents. In RBAC systems, including EGRBAC, we could construct different user roles for different token values for each teenager. However, this may result in having so many users' roles. Moreover, no mechanism exists to dynamically activate and deactivate specific user roles according to the current users' assigned tokens. To reduce the role explosion, we could define a single role for each token value for all teenagers. However, we would also need a mechanism for the RBAC model to dynamically activate or deactivate users' membership in different roles according to their current tokens' values. A similar situation arises in EGRBAC when we deal with dynamic device attributes, where increasing numbers of devices and dynamic attributes will lead to the explosion of device roles. See supplemental material for more information, available online.

Unlike traditional computing systems, IoT systems have dynamic nature. The dynamism of communication between people, connected devices, data, utility, and the changing nature of the system and environment characteristics in smart IoT connected systems requires actors' rights and access requirements to change accordingly. Hence, it is critical to have an access control model that captures different static and dynamic users', devices', and environmental characteristics.

2. Access administration tasks are simpler in RBAC models than in ABAC models.

In RBAC models, including EGRBAC, determining the role structure could take much effort, but when completed, access review is an easy task. It is easy to define who has what permissions by looking into a user's roles. On the other hand, in ABAC-based models, including HABAC, to determine the permissions available to a particular user, a large set of rules might need to be executed in exactly the same order the system applies them. This can make it practically impossible to determine risk exposure for a given user, as noted by [14]. Users provisioning is easier in RBAC-based models (including EGRBAC) than in ABAC-based models (including HABAC). In RBAC models, users or devices provisioning requires the administrator (the homeowner) to assign users' roles or devices' roles to newly created users or devices, respectively. Alternatively, in ABAC-based models, the administrator must configure different attribute values for newly provisioned users and devices. Moreover, in RBAC-based models changing the set of permissions available to any user require changing the set of roles assigned to that user. However, changing the set of permissions available to any user in ABAC-based models is a more complicated task. Generally speaking, RBAC-based models trade-up front role structuring effort for ease of administration and user permission review, while ABAC-based models make the reverse trade-off: it is easy to set up, but analyzing or changing user permissions can be problematic.

3. HABAC cannot prescribe limits on the set of permissions available for each user

In EGRBAC, we have three types of constraints: (1) Static Separation of Duty, (2) Dynamic Separation of Duty, and (3) Permission-role constraint [15]. On the other hand, in HABAC, we have two constraints: (1) Constraints on user attributes, equivalent to the static separation of duty, and (2) Constraints on session attributes, equivalent to the dynamic separation of duty. However, ABAC-based models, including HABAC, cannot handle permission-role constraints [10].

Permission-role constraints prevent specific users' roles from being capable of getting access to specific permissions at assignment time. This constraint allows system administrators to prevent future assignments that enable specific roles to get access to specific permissions. In HABAC, on the other hand, we cannot create something equivalent to EGRBAC permission role constraints. The main reason behind that is that in EGRBAC, the way a user get access to a specific set of permissions happens through a series of assignments. The most critical assignment is the role pair device role assignment ($RPDRA$), which assigns role pairs to device roles. Hence, by controlling $RPDRA$, we can control which role pairs and hence roles get access to which device roles and hence permissions. In HABAC, we do not have a similar "choke point" that can be controlled to prevent inadvertent or malicious assignments that may lead to unwanted access rights. This is a significant advantage of EGRBAC, where we can enforce such constraints at assignment time. See supplemental material for more information, available online.

From the above, a valid question is: Can we combine these two models to combine their advantages while eliminating their disadvantages? Developing hybrid models combining HABAC and EGRBAC features may be the most suitable for smart home IoT. Moreover, the high dynamism nature of smart home IoT is common in the majority of IoT application domains. Hence, hybrid models are likely the most suitable for IoT systems in general. The authors in [14] suggested three different approaches to combine RBAC [20] and ABAC [21] in a brief and high level way. These approaches are the dynamic roles approach, the attribute-centric approach, and the role-centric approach. However, they did not formalize these approaches into formal policy models nor implement or test them.

Inspired by the role-centric approach and the attribute-centric approach, in this research, we proposed two approaches to combine EGRBAC and HABAC: (a) Role-centric approach to build HyBAC_{RC} on top of EGRBAC. (b) Attribute-centric approach to build HyBAC_{AC} on top of HABAC. The reasons behind using two different approaches to develop two models are as follow:

1) If an RBAC-based model is already implemented, model designers can extend it using the role-centric approach without replacing the entire model. On the other hand, if an ABAC-based model is already implemented, model designers can extend it using the attribute-centric approach without the need to replace the entire system.
2) To provide policy model designers with two approaches for building hybrid models with the same expressiveness power. Choosing between them will be a trade-off between considerable front role structuring effort for ease of administration and access review in the role-centric model, on the one hand, and between easy setup effort but more complicated administration and access review tasks in the attribute-centric model on the other hand.

The basic idea in the approach of the dynamic roles is to use dynamic attributes by a front-end module to determine the user's role, retaining a conventional role structure but changing role sets dynamically. The reason for avoiding the dynamic role approach is that the home IoT environment is rich with attributes. Having many combinations of attributes' values may result in large numbers of user roles and device roles, further complicating the model and implementation. Such an approach may better fit environments with few dynamic attributes.

It is true that these models (EGRBAC, HABAC, HyBAC_{RC}, and HyBAC_{AC}) are proposed for the smart home IoT use case. However, they can be adapted and implemented in any IoT environment. Indeed, there is no restriction in using these models outside the home IoT use case scope. Hence, having a family of access control models ranging from relatively

simple to more sophisticated with better features and more expressiveness power will provide policy designers with a range of models to choose from according to the environment and the business requirements.

The main contributions of this paper are as follows.

- We motivate the need for hybrid models that combine ABAC and RBAC components to better capture smart home IoT access control requirements in specific and other IoT application domains in general.
- We propose two hybrid models for smart home IoT access control. In developing these models, we followed a role-centric approach to develop HyBAC$_{RC}$ and an attribute-centric approach to develop HyBAC$_{AC}$. Each model is formally defined and illustrated with a use case.
- To verify the applicability of the HyBAC$_{RC}$ and the HyBAC$_{AC}$ models using commercially available systems, we provide proof of concept implementation for each model.
- We conduct a comprehensive theoretical comparison between the HyBAC$_{RC}$ model, the HyBAC$_{AC}$ model, the EGRBAC model, and the HABAC model, where EGRBAC and HABAC are access control models previously developed to meet smart home IoT requirements.

2.1 Threat Model

In this paper, our threat model is the insiders with legitimate digital and physical access to the house, such as family members, guests, and workers. Our goal is to ensure that legitimate users get access only to what they are authorized to by the house owner.

3 Related Work

IoT technology has been investigated by many security researchers to identify its security and privacy vulnerabilities and to investigate design issues in different IoT frameworks, as in [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], and [29], [33], [34], [35], [36], [37] respectively. Access control in IoT is one of the most critical security services that mostly all researchers agree upon. Ouaddah et al. [38] provide an extensive survey on access control in IoT environments. The authors in [39], [40], [41], [42] also provided surveys on IoT applications access control models, challenges, and requirements. The rest of this section analyzes IoT access control models from the literature. The models are categorized according to their foundational model, viz., RBAC, ABAC, UCON, and CapBAC.

IoT Access Control Models Based On RBAC

The basic concept of the role-based access control (RBAC) model [20], [43] is that permissions are associated with roles, and users are made members of appropriate roles, thereby acquiring the roles' permissions. In [44], the authors extended RBAC by introducing context constraints. However, they mainly focused on the environment of web services. Researchers in [18], [45] proposed two different solutions, but both of them are focused on Web of Things [46], [47]. Their models are not adequate for smart homes. In the first solution, the architecture is completely centralized in a central access control decision facility coupled with a database. Access control decisions are taken outside the house, requiring a live connection and increasing the attack surface. 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 the 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. The RBAC model for IoT was also adopted in [48], [49]. However, the authors focus on providing an authentication protocol, while they only gave a high level overview of their RBAC model. Many works adapt the RBAC model to IoT [6], [50]. However, the use of resource-constrained devices is rarely addressed. Moreover, as we mentioned in Section 2, RBAC-based models cannot handle dynamic attributes.

IoT Access Control Models Based on ABAC

Different attribute-based access control (ABAC) models have been proposed in the literature (e.g., [21], [51]). In ABAC, access is granted according to attributes associated with the user and resource. In [52], and [53] the authors proposed sophisticated attribute-based encryption (ABE) models for smart grids. However, they did not discuss the ABAC models that they considered. Moreover, the ABE model may not be appropriate for computationally constrained devices as in smart things. In [54], the authors provided an attribute-based signatures approach to support anonymous access control. In [55], the authors introduced a formalized dynamic and fine-grained ABAC model for the smart car use cases. Bhatt et al. [56] proposed a conceptual attribute-based access control and communication control model for IoT. However, their access control model does not capture environment attributes.

IoT Access Control Models Based on CapBAC

Capability-based access control (CapBAC) utilizes the concept of capability, first introduced in [57], as a token 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 of the CapBAC model is that it requires all devices to implement CapBAC, which is unlikely given the heterogeneity of the 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 [38] give a survey on solutions proposed using the CapBAC model.

IoT Access Control Models Based on UCON

The distinctive properties of usage control (UCON) beyond traditional ABAC are the continuity of access decisions and the mutability of subject and object attributes [58], [59], [60]. A few solutions have been proposed in the literature based on UCON. However, these models cannot be adopted yet for various reasons. In [61], the model is proposed as a Device to Services (D-S) access control model. Moreover, no implementation was provided; only two theoretical experiments were introduced and assessed. In [62], 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 [63], 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., [64], [65], [66]. However, as [65] described, blockchain technology has some technical characteristics that could limit its applicability. First, cryptocurrency fees are a fundamental part of blockchain-based platforms, and all transactions include a fee. Second, transactions take time to get accepted into the blockchain, impacting processing time.

IoT Access Control Models Based on Other Models

In the literature, several other access control models for IoT have been proposed. For instance, in [67] the authors proposed a certificate-based device access control model in an IoT environment. Researchers in [17], [38], [39], [40], [41] have conducted surveys on different IoT access control models in the literature.

Recent research by [2] outlined a new perspective on home access control policies. According to them, smart home IoT has unique characteristics that require rethinking access control. Nevertheless, very few IoT solutions are specifically designed to meet smart home IoT requirements. Here are some examples. ABAC access control framework for smart homes is described in [5]. However, use cases and performance evaluations are lacking. The authors in [68] developed GRBAC, which introduced the notion of environment and device roles to capture environmental conditions and to enable device categorization, respectively, but did not give a formal model. Subsequently, they provided a brief but incomplete formalization without implementation [69]. Furthermore, in [70], an identity-based encryption model was described to implement a function-based access control model in smart homes. Authors of [71] presented a protocol to secure and authenticate smart homes. In addition, some researchers [72] have created a multi-user, multi-device access control mechanism for a smart home. The authors in [73] considered the ABAC model presented in [21] and provided an enforcement architecture model for smart home IoT. Based on He et al analysis [2] and Ouddah et al survey [38], the authors in [15] recently proposed a criteria for home IoT access control models. Accordingly, they proposed the EGRBAC model. It is an RBAC-based policy model for smart home IoT access control that complies with both[2], and [15] characteristics. Comparing EGRBAC to traditional RBAC, it incorporates contextual environmental changes and different device and permission characteristics. Hence, it rebuts the argument that RBAC-based models are unsuitable when handling the environment and device or permissions characteristics. However, their model can not capture dynamic attributes. Recently, Ameer et al. [19] provided an ABAC-based access control model for smart home IoT. However, as mentioned in Section 2 ABAC-based models cannot limit the permissions available for each user. Moreover, administration tasks are more complicated in ABAC-based models than in RBAC-based models. Few hybrid access control models that combine ABAC and RBAC features were developed in the literature [8], [9], [74], [75]. However, non of them developed explicitly to meet smart home IoT challenges. Furthermore, as [76] described, none of them provided an implementation for their model. Moreover, except for [74] they all followed the dynamic role approach. Recently, some researches suggested the use of activity-centric access control [77], [78], [79] and the score-based access control [80] in IoT application domains. However, these ideas are still in their early preliminary stages and have not yet formed into actual policy models.

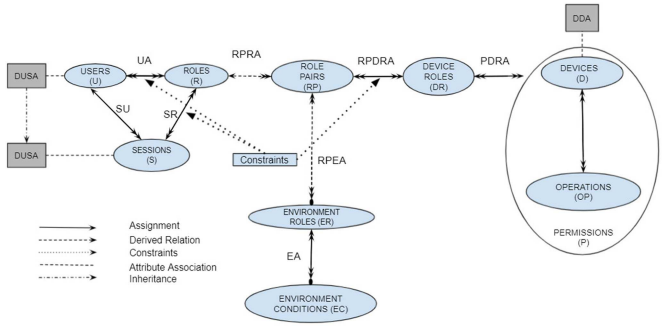

Fig. 1. Smart home IoT HyBAC$_{RC}$ model.

4 HyBAC$_{RC}$ Model

This section introduces the HyBAC$_{RC}$ model. The model is conceptually depicted in Fig. 1 while formal definitions are given in Tables 1 and 2. HyBAC$_{RC}$ is a hybrid model that is built on top of EGRBAC. We followed a role-centric approach in developing HyBAC$_{RC}$. In general, we have two types of attributes that capture different users, devices, and environment characteristics. These types are static attributes and dynamic attributes. Static Attributes tend to remain static (they evaluate to the same values) over a long period. Setting and changing the values of static attributes may require administrator intervention, for example, the user relationship to the house, the danger level of the device, etc. On the other hand, dynamic attributes are constantly changing due to various circumstances, possibly rapidly and unpredictably, for instance, user location, device temperature, etc. Values of dynamic attributes are automatically determined by sensors deployed in the smart home and under homeowner control. In this role-centric approach, relatively static attributes (for users, sessions, and devices) define the user's and device's roles. On the other hand, dynamic attributes define attribute-based rules within the scope of the relatively static roles. In this way, we avoid costly designs that might result from purely using roles or attributes. For example, consider a system with ten user attributes, four static and six dynamic. In the worst case, this could result in 2^{10} roles in RBAC or 2^{10} rules in ABAC. Establishing a policy structure based on the four static and six dynamic attributes amounts to a worst-case of 16 roles and 64 rules while effectively separating the policy concerns of the relatively static and intrinsically dynamic attributes. Moreover, using this hybrid design solves both problems of role explosion and the need for a dynamic role activation and deactivation mechanism when trying to capture dynamic attributes in role-based access

TABLE 1
$HyBAC_{RC}$ Model Formalization Part I: Basic Sets and Dynamic Attributes

Users, Roles and Sessions
- U and R are sets of users and roles respectively (home owner specified)
- $UA \subseteq U \times R$, many to many user role assignment relation (home owner specified)
  We define the derived function $roles(u) : U \rightarrow 2^R$, where: $roles(u_i) = \{r_j \mid (u_i, r_j) \in UA\}$
- S is the set of sessions (each session is created, terminated and controlled by an individual user)
- $SU \subseteq S \times U$, many to one relation assigning each session to its single controlling user
  We define the derived function $user(s) : S \rightarrow U$, where: $user(s_i) = u_j$ such that $(s_i, u_j) \in SU$
- $SR \subseteq S \times R$, many to many relation that assigns each session to a set of roles that can be changed by the controlling user
  We define the derived function $roles(s) : S \rightarrow 2^R$, where: $roles(s_i) = \{r_j \mid (s_i, r_j) \in SR\}$
  It is required that $roles(s) \subseteq roles(user(s))$ at all times

Devices, Operations, Permissions and Device Roles
- D is the set of devices deployed in the smart home (home owner deployed)
- OP and $P \subseteq D \times OP$ are sets of operations and permissions respectively (device manufacturers specified)
- DR is the set of device roles (home owner specified)
- $PDRA \subseteq P \times DR$, many to many permissions to device roles assignment (home owner specified)
  We define the derived function $droles(p) : P \rightarrow 2^{DR}$, where: $droles(p_i) = \{dr_j \mid (p_i, dr_j) \in PDRA\}$

Environment Conditions and Environment Roles
- EC is the set of boolean environment conditions (determined by sensors deployed in the smart home under home owner control)
  At any moment each $ec_i \in EC$ is either True or False depending on the state of the corresponding sensor
- ER is the set of environment roles (home owner specified)
- $EA \subseteq 2^{EC} \times ER$, many to many environment role activation relation (home owner specified)
  At any moment, $er \in ER$ is activated iff $(\exists(ec_{i1}, ec_{i2}, \ldots, ec_{in}), er) \in EA)[ec_{i1} \wedge ec_{i2} \wedge \ldots \wedge ec_{in} = \text{True}]$ at that moment

Role Pairs
- $RP \subseteq R \times 2^{ER}$, many to many role pairings of user role and subsets of environment roles (home owner specified)
  For $rp = (r_i, ER_j) \in RP$, we define $rp.r = r_i$ and $rp.ER = ER_j$
  We define the derived relation $RPRA \subseteq RP \times R$ where: $RPRA = \{(rp_m, r_n) \mid rp_m \in RP \wedge rp_m.r = r_n\}$
  We define the derived relation $RPEA \subseteq RP \times 2^{ER}$ where: $RPEA = \{(rp_m, ER_n) \mid rp_m \in RP \wedge ER_n = rp_m.ER\}$

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

Constraints
- $PRConstraints \subseteq 2^P \times 2^R$, many to many permission-role constraints relation (home owner specified)
  For each $(P_i, R_j) \in PRConstraints$ it is required that
  $(\forall p_m \in P_i)(\forall r_n \in R_j)(\forall (rp_p, dr_q) \in RPDRA)[(p_m, dr_q) \notin PDRA \vee rp_p.r \neq r_n]$
- $SSDConstraints \subseteq R \times 2^R$, many to many static separation of duty constraints relation (home owner specified)
  For each $(r_i, R_j) \in SSDConstraints$ it is required that $(\forall u \in U)(\forall r \in R_j)[(u, r) \in UA \implies (u, r_i) \notin UA]$
- $DSDConstraints \subseteq R \times 2^R$, many to many dynamic separation of duty constraints relation (home owner specified)
  For each $(r_i, R_j) \in DSDConstraints$ it is required that $(\forall s \in S)(\forall r \in R_j)[(s, r) \in SR \implies (s, r_i) \notin SR]$

Dynamic Attributes
- $DUSA$ and DDA are finite sets of dynamic user and session attribute functions and dynamic device attribute functions respectively, where $DUSA \cap DDA = \emptyset$ for convenience (determined by sensors deployed in the smart home under home owner control)
- Each session s inherits a subset of the dynamic attribute functions of its unique user creator (controlled by the session creator $user(s)$)
  For every inherited attribute function $att \in DUSA$, $att(s) = att(user(s))$ at all time
  For every non-inherited attribute function $att \in DUSA$, $att(s)$ is undefined and its use in any logical formula renders that formula false
- For each $att \in DUSA \cup DDA$, $Range(att)$ is the attribute range which is a finite set of atomic values
- $attType : DUSA \cup DDA \rightarrow \{set, atomic\}$.
- Each $att \in DUSA \cup DDA$ correspondingly maps users in U or sessions in S, or devices in D to atomic or set attribute values. Formally:

$$att : U \text{ or } S \text{ or } D \rightarrow \begin{cases} Range(att), & \text{if } attType(att) = atomic \\ 2^{Range(att)}, & \text{if } attType(att) = set \end{cases}$$

  At any moment, the value of att for a given user or device is automatically determined by sensors deployed in the smart home

Attributes Authorization Function
- $Authorization(s : S, op : OP, d : D)$ is a logic formula defined using the grammar of Table 2 (home owner specified)
  It is evaluated for a specific session s_i, device d_j and operation op_k as specified in Table 2

CheckAccess Predicate
- $CheckAccess$ is evaluated when session s_i attempts operation op_k on device d_j while the environment conditions in EC_l are True
- $CheckAccess(s_i, op_k, d_j, E_l)$ evaluates to True or False using the following formula:

$$Authorization(s_i, op_k, d_j) \wedge$$
$$(\exists (rp_m, dr_n) \in RPDRA)\ [((d_j, op_k), dr_n) \in PDRA \wedge (s_i, rp_m.r) \in SR \wedge$$
$$rp_m.ER \subseteq \{er \in ER \mid (\exists EC'_l \subseteq EC_l)[(EC'_l, er) \in EA]\}]$$

TABLE 2
HyBAC$_{RC}$ Model Formalization Part II: Attributes Authorization Function

Attributes Authorization Function

- $Authorization(s : S, op : OP, d : D)$ is a propositional logic formula returning true or false specified using the following grammar.
  - $\alpha ::= term \mid term \wedge term \mid term \vee term \mid (term) \mid \neg term \mid \exists x \in set.\alpha \mid \forall x \in set.\alpha$
  - $term ::= setsetcompareset \mid atomic \in set \mid atomic \notin set \mid atomicatomiccompareatomic$
  - $setcompare ::= \subset \mid \subseteq \mid \not\subseteq$
  - $atomiccompare ::= < \mid = \mid \leq$
  - $set ::= dusa(s) \mid dda(d) \mid roles(s) \mid droles((op, d))$, for $attType(dusa) = set$ and $attType(dda) = set$
  - $atomic ::= dusa(s) \mid dda(d) \mid value$, for $attType(dusa) = atomic$ and $attType(dda) = atomic$
- For a specific session s_i, device d_j and operation op_k, the authorization function $Authorization(s_i, op_k, d_j)$ is evaluated by substituting the actual attribute values of $dusa(s_i)$, $dda(d_j)$, $roles(s_i)$ and $droles((op_k, d_j))$ for the corresponding symbolic placeholders and evaluating the resulting logical formula to be True or False. Any term that references an undefined attribute value is evaluated as False

control models (including the EGRBAC model), as discussed in Section 2. The basic components of HyBAC$_{RC}$ are discussed below.

Basic Sets and Assignmnets. Users (U) set refers to humans interacting with smart home devices as authorized. Roles (R) are similar to the traditional RBAC user's roles. In smart homes, however, the role is a way to express the relationship between the user and the house's family, including parents, kids, teenagers, babysitters, etc. The many-to-many UA relation specifies that one user can have more than one role. An example of a user with two different roles is a neighbor who also is a plumber who needs temporary access to repair an appliance and therefore should have different privileges as a worker. Sessions (S) are created by users. Each user creates one or more sessions during which he may activate a subset of his assigned roles. A user might have multiple sessions active concurrently and asynchronously. Each session is linked to a unique, controlling user through the many to one SU relation. Each session is mapped to a set of associated roles through the many to many SR relation. Smart home devices are represented by the set of devices (D). The operation (OP) set refers to the actions allowed to be performed on devices as specified by device manufacturers. The approval of an operation to be performed on a device is called permission, which is a device operations pair. The permissions set (P) is a subset of $D \times OP$. The set device roles (DR) represents a mean for categorizing permissions of different devices. For example, we can categorize the dangerous permissions of various smart devices by creating a device role called dangerous devices and assigning dangerous permissions (such as turning on the oven, turning on the gas stove, and opening and closing the front door lock) to it. Assigning different permissions to different device roles is captured through the many-to-many set of assignments $PDRA$. Environment roles (ER) represent environmental circumstances, such as daytime/nighttime. Environment roles are turned on/off (i.e., triggered) by subsets of Environment Conditions (EC) such as daylight. These environmental conditions must be active together to trigger a specific environmental role through the EA relation. The role pairs (RP) set represents different combinations of roles and environment roles' subsets. Each role pair rp has a role part $rp.r$, the single role associated with rp, and an environment role part $rp.ER$, the subset of the environment roles associated with rp. Since some ER subsets may not be meaningful, the permissible role pairs RP are defined as a subset of $R \times 2^{ER}$. Each role has one or more role pairs associated with it through the relationship $RPRA$. Each role pair is associated with a subset of ER through $RPEA$. $RPDRA$ binds these components together by assigning device roles to role pairs. Accordingly, for each role pair rp, the single role associated with it through $RPRA$ has access to all device roles assigned to it through $RPDRA$ when the set of environment roles associated with it through $RPEA$ are active.

Derived Functions. There are three derived functions that are useful to define. First, the function $roles(s)$ which takes a session s as an input and returns the set of roles assigned to s. Second, the function $users(s)$ takes a session s as an input and returns the unique user who created s (constant for the session lifetime). Third, the function $droles(p)$ takes permission p as an input and returns the set of device roles to which the permission p is assigned.

In HyBAC$_{RC}$, at any specific moment and depending upon the current active roles in a session that belong to a user u and the current active environment conditions (which determine the current active environment roles), some role pairs are active. Hence, the maximum set of permissions that may currently be available for the user u is equal to the set of permissions assigned to the device roles which are assigned to the current active role pairs. However, which permissions among these maximum permissions are currently available for this user is further determined by the current values of the dynamic user and session attributes and the current values of the dynamic device attributes.

Dynamic Attributes. These components are shown as grey rectangles in Fig. 1. Attributes are characteristics that are used in access control decisions. An attribute is a function that takes an entity, such as a user, and returns a specific value from its range. A finite set of atomic values gives an attribute range. An atomic valued attribute will return one value from the range, while a set valued attribute will return a subset of the range. As discussed earlier, dynamic attributes change due to different rapidly changing conditions. An example of users' dynamic attributes can be user location, user temporal health condition, user waking status, etc. Similarly, we may be interested in device location, temperature, usage status, etc. Dynamic user and session attribute functions ($DUSA$) are the set of attributes associated with both users and sessions. Each session s inherits a subset of its unique user creator's dynamic attributes. If a session s inherits a dynamic user session attribute $dusa$ from his user creator $user(s)$, then it is required that $dusa(s) = dusa(user(s))$. How the inheritance process happens is outside the scope of HyBAC$_{RC}$ operational model and can be considered as part of an administrative model. The dynamic device attribute functions set (DDA) is the set of dynamic attributes associated with devices. In HyBAC$_{RC}$, we do not consider environment and operations dynamic attributes for the following reasons. First, regarding the element environment roles (ER), the way it is designed and associated with the role pairs (RP) element enables it to capture both static and dynamic environment attributes. A role pair will be active only if the set of associated environment roles are currently active and triggered by their corresponding environment conditions. Second, operations (OP) are basically defined by the manufacturer and usually have a static nature, such as dangerous or benign.

Attributes Authorization Function (Rules). An attribute authorization function is a logical formula evaluated for each access decision. It is defined using the grammar of Table 2. For a specific session s_i, device d_j and operation op_k the authorization function $Authorization(s_i, op_k, d_j)$ is evaluated by substituting the actual attribute values of $dusa(s_i)$, $dda(d_j)$, $roles(s_i)$ and $droles((op_k, d_j))$ for the corresponding symbolic placeholders and evaluating the resulting logical formula to be True or False. The output of this function defines whether the requesting session s_i is authorized to perform the requested operation op_k on the requested device d_j at the current instance of time according to the current dynamic attributes values of s_i and the current dynamic attributes values of d_j.

Check Access Predicate. The bottom part of Table 1 formalizes the check access predicate of HyBAC$_{RC}$. Consider a session s_i which attempts to perform operation op_k on device d_j when the subset of environment conditions EC_l are active. The access will be granted if and only if both of the following are true:

1) The maximum set of permissions available for the session s_i allows it to perform the operation op_k on the device d_j according to the role membership and role activation requirements. These are specified in the last two lines of the authorization predicate in Table 1. The role membership requirements can be stated in words as follows. There is a role pair rp_m and a device role dr_n assigned to each other in $RPDRA$ such that the following conditions are true: (i) dr_n is assigned to the permission (d_j, op_k) in $PDRA$, (ii) $rp_m.r$ is one of the active roles of s_i (as given in SR), and (iii) each environment role $er \in rp_m.ER$ is active because it is activated by a subset of the currently active environment conditions EC_l.
2) The authorization function $Authorization(s_i, op_k, d_j)$ evaluates to TRUE. This indicates that s_i is allowed to perform op_k on d_j according to the current dynamic attributes values of s_i and d_j.

Constraints. Constraints are invariants that must never be violated. In HyBAC$_{RC}$, we have three types of constraints, as shown in the following.

Permission-Role Constraint. These constraints prevent $RPDRA$ assignments that would enable specific roles to access specifically prohibited permissions. Formally, $PRConstraints \subseteq 2^P \times 2^R$ constitute a many to many subsets of permissions to subsets of roles relation. The basic idea in $PRConstraint$ is that: For every (P_i, R_j) in the set of $PRConstraints$, and for every $p_m \in P_i$ and $r_n \in R_j$, it is forbidden to assign any device role dr_x that the permission p_m is assigned to, to any role pair rp_y with r_n as the role part of it (formally $(rp_m, dr_x) \notin RPDRA$).

Static Separation of Duty (SSD). It is the familiar SSD in RBAC. It constrains the assignment of roles to users. Thus, if a user is authorized to be a member of one role, he or she is not authorized to be a member of another conflicting role [81].

Dynamic Separation of Duty (DSD). It is the familiar DSD in RBAC. DSD allows users to be members of roles that do not present a conflict of interest when acted independently as part of different sessions but create conflicts when acted upon simultaneously in the same session [81].

HyBAC$_{RC}$ is an operational access control model. Managing and enforcing different types of constraints can be considered part of an administrative access control model, which is outside the scope of this manuscript.

4.1 Use Case Demonstration

Here we present one use case scenario to demonstrate how to configure HyBAC$_{RC}$ components to enforce specific access control policies. This use case has the following objectives. (a) Authorize teenagers to use dangerous permissions in kitchen devices (open the oven and turn on the oven) only when a parent is in the kitchen and the current temperature of the oven is less than or equal to $150°$. (b) Authorize teenagers to unconditionally use nondangerous permissions in kitchen devices (close the oven, turn off the oven, open and close the fridge). (c) Authorize teenagers to use the front door lock permissions (lock and unlock) if they are temporarily granted those permissions by one of the parents. (d) Allow teenagers to use entertainment devices permissions during weekends evenings and nights if the accessed device is not in use by someone else. (e) Allow kids to use kids friendly operations in entertainment devices which are, turning on and off the device and rated G content only during weekend evenings and if the accessed device is not in use by someone else. (f) Finally, allow parents to use any operation in any device unconditionally. For this use case, HyBAC$_{RC}$ will be configured as shown in Fig. 2.

First, we configure the maximum permissions available for each user. We have five users bob, $alex$, $suzanne$, $john$, and $anne$ respectively assigned to roles $parents$, $kids$, $kids$, $teenagers$, and $teenagers$. The devices include $Oven, Fridge$, $FrontDoorLock$, $PlayStation$, and TV. The device $Oven$ has four operations, turning on the oven On_{Oven}, turning off the oven Off_{Oven}, open the oven $Open_{Oven}$, and close the oven $Close_{Oven}$. The device $Fridge$ has three operations, open the fridge $Open_{Fridge}$, close the fridge $Close_{Fridge}$, and check the fridge temperature $Check\_temperature_{Fridge}$. The device $FrontDoorLock$ has two operations, lock $Lock_{FrontDoorLock}$ and unlock $Unlock_{FrontDoorLock}$. The $PlayStation$ device has two operations, turning on On_{PS} and turning off Off_{PS}. Finally, the TV device has four operations, turning on On_{TV}, turning off Off_{TV}, rated G content G_{TV}, rated PG content PG_{TV}, and rated R content R_{TV}. We have five device roles. We assign the oven permissions $\{On_{Oven}, Open_{Oven}\}$ to the device role $Dangerous\ \_Kitchen\_Permissions$. We assign the oven permissions $\{Off_{Oven}, Close_{Oven}\}$ and all fridge permissions to the $Non\_Dangerous\_\ Kitchen\_Permissions$ device role. Moreover, we assign all front door lock permissions to the device role $Front\_Door\_Lock$. All the permissions of the TV and the $PlayStation$ devices are assigned to $Entertainment\_Devices$ device role, and an appropriate subset of these permissions are assigned to $Kids\_Friendly\_Content$ device role. EC comprises $Parent\_Is\_In\_The\_Kitchen$, $weekends$, $evenings$, $nights$, and $TRUE$, respectively active when a parent is in the kitchen, on weekends, during the evening, during the night, and always. The environment role $Teenagers\_Kitchen\_Time$ is active when the environment condition $Parent\_Is\_In\_The\_Kitchen$ is active. The environment role $Kids\_\ Entertainment\_Time$ is active when both environment conditions $weekends$ and $evenings$ are active. Similarly, the environment role $Teenagers\_\ Entertainment\_\ Time$ is active when $weekends$ and $evenings$ or $weekends$ and $nights$ environment conditions are active. The environment condition Any_Time is always active.

$RPDRA$ assignments specify the maximum permissions available for each role according to the device roles assignments and the current active environment roles. The first assignment indicates that $kids$ can access the permissions assigned to the device role $Kids\_Friendly\_Content$ when $Kids\_Entertainment\_Time$ is active. The second assignment shows that $teenagers$ can access $Entertainment\_Devices$ device role when the environment conditions $Teenagers\_\ Entertainment\_Time$ is active. The third assignment indicates that $teenagers$ can access the permissions assigned to $Dangerous\_Kitchen\_\ Permissions$ when $Teenagers\_$

$U = \{bob, alex, suzanne, john, anne\}$, $R = \{parents, kids, teenagers\}$
$UA = \{(bob, parents), (alex, kids), (suzanne, kids), (anne, teenagers), (john, teenagers)\}$

$D = \{Oven, Fridge, FrontDoorLock, PlayStation, TV\}$, $OP = OP_{Oven} \cup OP_{Fridge} \cup OP_{FrontDoorLock} \cup OP_{PlayStation} \cup OP_{TV}$, where
$OP_{Oven} = \{On_{Oven}, \mathit{Off}_{Oven}, Open_{Oven}, Close_{Oven}\}$, $OP_{Fridge} = \{Open_{Fridge}, Close_{Fridge}, Check\_temperature_{Fridge}\}$,
$OP_{FrontDoorLock} = \{Lock_{FrontDoorLock}, Unlock_{FrontDoorLock}\}$, $OP_{PlayStation} = \{On_{PS}, \mathit{Off}_{PS}\}$,$OP_{TV} = \{On_{TV}, \mathit{Off}_{TV}, G_{TV}, PG_{TV}, R_{TV}\}$
$P = P_{Oven} \cup P_{Fridge} \cup P_{FrontDoorLock} \cup P_{PlayStation} \cup P_{TV}$, where
$P_{Oven} = \{Oven\} \times OP_{Oven}$, $P_{Fridge} = \{Fridge\} \times OP_{Fridge}$, $P_{FrontDoorLock} = \{FrontDoorLock\} \times OP_{FrontDoorLock}$,
$P_{PlayStation} = \{PlayStation\} \times \{On, \mathit{Off}\}$, $P_{TV} = \{TV\} \times \{On_{TV}, \mathit{Off}_{TV}, G_{TV}, PG_{TV}, R_{TV}\}$
Let $P_1 = \{Oven\} \times \{On_{Oven}, Open_{Oven}\}$, $P_2 = \{Oven\} \times \{\mathit{Off}_{Oven}, Close_{Oven}\}$, $P_3 = \{TV\} \times \{On_{TV}, \mathit{Off}_{TV}, G_{TV}\}$,
$DR = \{Dangerous\_Kitchen\_Permissions, Non\_Dangerous\_Kitchen\_Permissions, Front\_Door\_Lock, Kids\_Friendly\_Content, Entertainment\_Devices\}$
$PDRA = P_1 \times \{Dangerous\_Kitchen\_Permissions\} \cup (P_2 \cup P_{Fridge}) \times \{Non\_Dangerous\_Kitchen\_Permissions\} \cup P_{FrontDoorLock} \times \{Front\_Door\_Lock\} \cup P_{TV} \times \{Entertainment\_Devices\} \cup (P_3 \cup P_{PlayStation}) \times \{Kids\_Friendly\_Content\}$

$EC = \{Parent\_Is\_In\_The\_Kitchen, Weekends, Evenings, Nights, TRUE\}$
$ER = \{Teenagers\_Kitchen\_Time, Kids\_Entertainment\_Time, Teenagers\_Entertainment\_Time, Any\_Time\}$
$EA = \{(\{Parent\_Is\_In\_The\_Kitchen\}, Teenagers\_Kitchen\_Time), (\{weekends, evenings\}, Kids\_Entertainment\_Time), (\{weekends, evenings\}, Teenagers\_Entertainment\_Time), (\{weekends, nights\}, Teenagers\_Entertainment\_Time), (TRUE, Any\_Time)\}$

$RP = \{(teenager, \{Teenagers\_Kitchen\_Time\}), (teenager, \{Teenagers\_Entertainment\_Time\}), (teenagers, \{Any\_Time\}), (kids, \{Kids\_Entertainment\_Time\}), (parents, \{Any\_Time\})\}$
$RPDRA = \{((kids, \{Kids\_Entertainment\_Time\}), Kids\_Friendly\_Contents),$
$((teenager, \{Teenagers\_Entertainment\_Time\}), Entertainment\_Devices),$
$((teenager, \{Teenagers\_Kitchen\_Time\}), Dangerous\_Kitchen\_Permissions),$
$((teenagers, \{Any\_Time\}), Non\_Dangerous\_Kitchen\_Permissions),$
$((teenagers, \{Any\_Time\}), Front\_Door\_Lock), ((parents, \{Any\_Time\}), Entertainment\_Devices),$
$((parents, \{Any\_Time\}), Non\_Dangerous\_Kitchen\_Permissions), ((parents, \{Any\_Time\}), Dangerous\_Kitchen\_Permissions),$
$((parents, \{Any\_Time\}), Front\_Door\_Lock)\}$

$PRConstraints = \{\{(Oven, On_{Oven}), (Oven, \mathit{Off}_{Oven}), (Fridge, Open_{Fridge}), (Fridge, Close_{Fridge})\}, \{kids\}\}$

$DUSA = \{Front\_Door\_Lock\_Token\}$, $DDA = \{Device\_Temperature, UsingStatus, UsingUser\}$
$Front\_Door\_Lock\_Token : u : U \rightarrow \{True, False\}$, $Front\_Door\_Lock\_Token : s : S \rightarrow \{True, False\}$
$Device\_Temperature : d : D \rightarrow \{x | x \text{ is an oven temperature}\}$, $UsingStatus : d : D \rightarrow \{True, False\}$, $UsingUser : d : D \rightarrow U$

$Authorization(s : S, op : OP, d : D) \equiv$
$(parents \in R(s)) \vee (teenager \in R(s) \wedge Dangerous\_Kitchen\_Permissions \in drole((op, d)) \wedge Device\_Temperature(d) \leq 150°) \vee$
$(teenager \in R(s) \wedge Non\_Dangerous\_Kitchen\_Permissions \in drole((op, d))) \vee$
$(teenager \in R(s) \wedge Entertainment\_Devices \in drole((op, d)) \wedge (\neg UsingStatus(d) \vee UsingUser(d) = user(s)) \vee$
$(teenager \in R(s) \wedge Front\_Door\_Lock \in drole((op, d)) \wedge Front\_Door\_Lock\_Token(s) = True) \vee$
$(kids \in R(s) \wedge Kids\_Friendly\_Contents \in drole((op, d)) \wedge (\neg UsingStatus(d) \vee UsingUser(d) = user(s))$

Fig. 2. HyBAC$_{RC}$ use case configuration.

Kitchen_ Time environment role is active, whereas the fourth assignment indicates that *teenagers* can access *Non_ Dangerous_Kitchen_Permissions* device role at any time. Moreover, the fifth assignment shows that *teenagers* can access *Front_ Door_ Lock* device role at any time. Finally, the last four assignments indicate that *parents* are authorized to access *Entertainment_Devices*, *Dangerous_ Kitchen_ Permissions*, *Non_ Dangerous_Kitchen_Permissions*, and *Front_ Door_ Lock* device roles at any time.

Next, we configure the dynamic attributes. We introduce one Boolean dynamic user and session attribute *Front_ Door_Lock_Token*, which is True when the homeowner has granted this user a token indicating that this user can access the front door lock device and False otherwise. The detailed mechanism by which the homeowner grants a temporal token for a specific user is outside the scope of the HyBAC$_{RC}$ operational model and can be considered as part of an administrative model. We also define three dynamic device attributes: (1) *Device_Temperature* whose value is determined by the device's temperature. (2) *UsingStatus* to indicate that a device is in use (True) or not (False). (3) *UsingUser*, which maps each device to the user currently using it (or is undefined if the device is not in use).

Finally, we define the authorization function as a disjunction of six conjunctive propositional clauses (rules). This authorization function further constrains the maximum permissions available for each user based on the dynamic attribute values. The first clause gives unconditional access to parents. The following two clauses define teenagers' authorization rules concerning dangerous and non-dangerous kitchen permissions. The second clause authorizes *teenagers* to access permissions belonging to the *Dangerous_Kitchen_ Permissions* device role if the permission device has a temperature $\leq 150°$. The third authorization rule gives *teenagers* access to *Non_ Dangerous_Kitchen_Permissions* unconditionally. The fourth authorization rule specifies that *teenagers* can access *Entertainment_Devices* device role permissions only if another user does not use the requested device. The fifth authorization rule specifies that *teenagers* can access *Front_ Door_ Lock* device role permissions only if they have the front door lock token indicating that they were allowed to access the front door lock permissions by one of the parents. Finally, the last clause specifies that *kids* can access the permissions assigned to the *Kids_Friendly_Content* device role if the requested permission's device is not used by someone else.

5 HyBAC$_{AC}$ Model

This section introduces the HyBAC$_{AC}$ model. Fig. 3 conceptually depicts the model components. The model formal definition is given in Tables 3 and 4. HyBAC$_{AC}$ model is a hybrid access control model that is built on top of HABAC. We followed an attribute-centric approach in developing

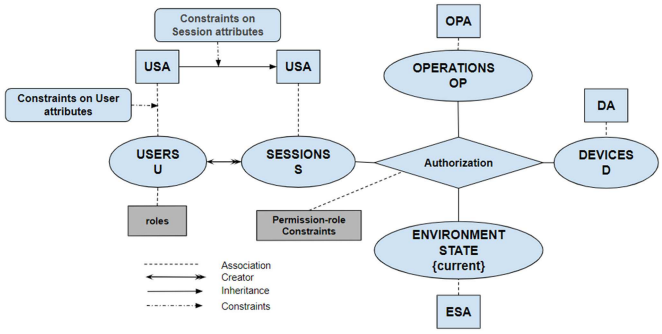

Fig. 3. Smart home IoT $HyBAC_{AC}$ model.

HyBAC$_{AC}$, where the user role is just one of many user attributes. In Section 2, we mentioned that one of the HABAC limitations is that it cannot limit the set of permissions available for each user as in EGRBAC. To solve this limitation in HyBAC$_{AC}$, we introduced the concept of Roles (R) (aka anti-roles), as we will explain in the following. The basic components of HyBAC$_{AC}$ are discussed below.

Basic Sets and Functions. Users (U) are humans interacting directly with smart things. Users create sessions (S) to perform some actions in the system. The session can only be terminated by the user who created it. The function $users(s)$ is used to associate each session s with its unique user creator. Environment States set, $ES = \{current\}$, is a singleton set that only includes the state "current". The current state is the picture of the environment in the present time instant. The component might be extended to include other instants of time, such as last week and last year, for example, $ES = \{current, lastweek, lastyear\}$. Devices ($D$) are smart home devices (smart things), such as smart door locks. Operation (OP) refers to the actions performed on devices following manufacturer specifications. Every device d is mapped to the set of valid operations on d by the function $ops(d)$.

Attributes are characteristics of users/sessions, devices, operations, and environment states which are used in access control decisions. Attributes are functions that take an entity, such as a user, and determine a specific value from its range. For each attribute function att_i, there is a range of

TABLE 3
HyBAC$_{AC}$ Model Formalization Part I: Basic Sets and Components

Basic Sets and Functions

- U is a finite sets of users (home owner specified)
- S is the set of sessions (each session is created, terminated and controlled by an individual user)
- The function $user(s) : S \rightarrow U$ maps each session to its unique creator and controlling user
- D is the set of devices deployed in the smart home (home owner deployed)
- OP is the set of possible operations on devices (device manufacturers specified)
- The function $ops : D \rightarrow 2^{OP}$ specifies the valid operations for each device (device manufacturers specified)
- $ES = \{current\}$ is a singleton set where $current$ denotes the environment at the current time instance

Attribute Functions and Values

- USA, DA, OPA and ESA are user/session, device, operation and environment-state attribute functions respectively, where for convenience we require USA, DA, OPA and ESA to be mutually exclusive
- Each session s inherits a subset of the attribute functions in USA from its unique user creator (controlled by the session creator $user(s)$). For every inherited attribute function $att \in USA$, $att(s) = att(user(s))$ at all time Unless otherwise specified use of a non-inherited session attribute in a logical formula renders that formula false
- For each attribute att in $USA \cup DA \cup OPA \cup ESA$, $Range(att)$ is the attribute range, a finite set of atomic values
- $attType : USA \cup DA \cup OPA \cup ESA \rightarrow \{set, atomic\}$.
- Each $att \in USA \cup DA \cup OPA \cup ESA$ correspondingly maps users in U/sessions in S, devices in D, operations in OP or the environment state $current$ to atomic or set attribute values. Formally:

$$att : U \text{ or } S \text{ or } D \text{ or } OP \text{ or } \{current\} \rightarrow \begin{cases} Range(att), & \text{if } attType(att) = atomic \\ 2^{Range(att)}, & \text{if } attType(att) = set \end{cases}$$

- Every $att \in USA \cup DA \cup OPA \cup ESA$, att is designated to be either a static or dynamic attribute where dynamic attributes must have corresponding sensors deployed in the smart home (under home owner control)
- Static attribute ranges and values are set and changed by administrator actions (by home owner or device manufacturers)
- Dynamic attribute ranges and values automatically determined by sensors deployed in the smart home (under home owner control) or set and changed by home owner.

Constraints

- $UAConstraint \subseteq UAP \times 2^{UAP}$ is the user attribute constraints relation (home owner specified) where $UAP = \{(usa, v) \mid usa \in USA \ \wedge \ v \in Range(usa))\}$

  Each $uac = ((usa_x, v_y), UAP_j) \in UAConstraint$ specifies the following invariant:

$$\begin{cases} (\forall u_l \in U)(\forall (usa_m, v_n) \in UAP_j)[usa_x(u_l) = v_y \Rightarrow usa_m(u_l) \neq v_n], & \text{if } attType(usa_x) = attType(usa_m) = atomic \\ (\forall u_l \in U)(\forall (usa_m, v_n) \in UAP_j)[v_y \in usa_x(u_l) \Rightarrow v_n \notin usa_m(u_l)], & \text{if } attType(usa_x) = attType(usa_m) = set \end{cases}$$

- $SAConstraint \subseteq UAP \times 2^{UAP}$ is the session attribute constraints relation (home owner specified)

  Each $sac = ((usa_x, v_y), UAP_j) \in SAConstraint$ specifies the following invariant:

$$\begin{cases} (\forall s_l \in S)(\forall (usa_m, v_n) \in UAP_j)[s_l \text{ inherits } usa_x \wedge usa_x(user(s_l)) = v_y \wedge usa_m(user(s_l)) = v_n \Rightarrow s_l \text{ does not inherit } usa_m], \\ \qquad\qquad \text{if } attType(usa_x) = attType(usa_m) = atomic \\ (\forall s_l \in S)(\forall (usa_m, v_n) \in UAP_j)[s_l \text{ inherits } usa_x \wedge v_y \in usa_x(user(s_l)) \wedge v_n \in usa_m(user(s_l)) \Rightarrow s_l \text{ does not inherit } usa_m], \\ \qquad\qquad \text{if } attType(usa_x) = attType(usa_m) = set \end{cases}$$

Attributes Authorization Function

- $Authorization(s : S, op : OP, d : D, current : ES)$ is a logic formula defined using the grammar of Table 4 (home owner specified)

  It is evaluated for a specific session s_i, operation op_k, device d_j and environment state $current$ as specified in Table 4

Roles (aka Anti-Roles)

- R is a finite set of roles (aka anti-roles) (home owner specified)
- The function $roles : U \rightarrow 2^R$ maps each user to a subset of roles (home owner specified)
- $PRConstraints \subseteq 2^P \times 2^R$, many to many permission-role constraints relation (home owner specified) where $P \subseteq D \times OP$ is a derived relation such that $(d, op) \in P \Leftrightarrow op \in ops(d)$

CheckAccess Predicate

- $CheckAccess$ is evaluated when session s_i attempts operation op_k on device d_j in context of environment state $current$
- $CheckAccess(s_i, op_k, d_j, current)$ evaluates to True or False using the following formula:

  $op_k \in ops(d_j) \ \wedge \ Authorization(s_i, op_k, d_j, current)) \wedge$
  $(\forall (P_x, R_y) \in PRConstraints)[((op_k, d_j) \notin P_x) \vee (roles(user(s_i))) \cap R_y = \phi]$

TABLE 4
HyBAC$_{AC}$ Model Formalization Part II: Attributes Authorization Function

Attributes Authorization Function

$-Authorization(s : S, op : OP, d : D, current : ES)$ is a propositional logic formula returning true or false specified using the following grammar.

- $\alpha ::= term \mid term \wedge term \mid term \vee term \mid (term) \mid \neg term \mid \exists x \in set.\alpha \mid \forall x \in set.\alpha$
- $term ::= set\ setcompare\ set \mid atomic \in set \mid atomic \notin set \mid atomic\ atomiccompare\ atomic$
- $setcompare ::= \subset \mid \subseteq \mid \not\subseteq$
- $atomiccompare ::= < \mid = \mid \leq$
- $set ::= usa(s) \mid opa(op) \mid esa(current) \mid da(d)$, where $attType(usa) = attType(opa) = attType(esa) = attType(da) = set$
- $atomic ::= usa(s) \mid opa(op) \mid esa(current) \mid da(d) \mid value$, where $attType(usa) = attType(opa) = attType(esa) = attType(da) = atomic$

$-$For a specific session s_i, device d_j and operation op_k the authorization function $Authorization(s_i, op_k, d_j, current)$ is evaluated by substituting the actual attribute values of $usa(s_i)$, $da(d_j)$, $opa(op_k)$ and $esa(current)$ for the corresponding symbolic placeholders and evaluating the resulting logical formula to be True or False
Any term that references an undefined attribute value is evaluated as False

possible values $Range(att_i)$ that att_i can be evaluated to. An atomic valued attribute will return one value from its range, while a set valued attribute will return a subset of its range. Different attributes, whether atomic or set valued attributes, can be categorized into two types: static and dynamic. User/Session attribute functions set (USA) is the set of attributes associated with both users and sessions. Sessions inherit a subset of the attributes of their unique creator. If a session s inherited a user session attribute usa from his creator user $user(s)$, then it is required that $usa(s) = usa(user(s))$. How the inheritance process happens is outside the scope of HyBAC$_{AC}$ operational model and can be considered as part of an administrative model. DA is a set of attribute functions related to smart devices, such as "kitchen devices", "ventilation devices", and "Bob's devices". OPA is the set of attributes associated with operations. For instance, we may want to characterize dangerous kitchen operations, so we create an operation attribute entitled "Dangerous Kitchen Operations" and associate it with those operations. Different environment characteristics such as "time" and "weather" are captured through the environment state attribute functions set (ESA). User/session attribute functions and environment state attribute functions are total functions, while operation attribute functions and device attribute functions, on the other hand, are partial functions (since we may have some devices or operations that are not assigned to some attributes).

Roles(R). R is a finite set of roles (aka anti-roles) specified by the homeowner. The homeowner assigns different roles to different users. The function $roles(u)$ maps each user $u \in U$ to a subset of roles (which are assigned to u by the homeowner). The name anti-roles came from the idea that these roles are defined to prevent unwanted access in permission-role constraints, as explained in the following.

Constraints. These are invariants that must never be violated. In HyBAC$_{AC}$ and similar to HyBAC$_{RC}$, we have three types of constraints as described below.

Permission-Role Constraints. HyBAC$_{AC}$ incorporates constraints that prevent specific users from accessing specific operations on specific devices. By assigning different users to different roles, permission-role constraints are then defined as a many to many permissions to role constraints relation. These constraints enable HyBAC$_{AC}$ to limit the set of permissions available for each user. For example, having a permission role constraint $prc_i \in PRConstraints$, where $prc_i = (\{(d_k, op_l)\}, \{r_m\})$ implies that a session s_i created by a user u_j, where $r_m \in roles(u_j)$, cannot access the operation op_l on the device d_k. Permission-role constraints are checked during execution time as part of the check access predicate. This is unlike the case in EGRBAC and HyBAC$_{RC}$ where the permission-role constraints are enforced at assignment time to prevent prohibited assignments. As we discussed in Section 2, HABAC is not capable of limiting the set of permissions available for different users; hence it cannot support this type of constraint.

Constraints on User Attributes. User's attributes' constraints represent the static separation of duty concept. It imposes restrictions on user attributes. If a specific attribute value is assigned to a user, the user is prohibited from being assigned to another attribute value. In defining this type of constraint, we use the user attribute pairs set UAP as shown in Table 3. UAP is a set of pairs. Each pair is a pair of a user attribute function and a user attribute function's value. For each pair (usa, v) in a UAP set. usa is an attribute function in the user/session attribute functions set, while v is a value in the usa attribute function range of values. This is formally defined as following: $UAP = \{(usa, v) \mid usa \in USA \wedge v \in Range(usa))\}$. For example, consider the following user attribute constraint

$$UAConstraint = \{uac_i\}$$
$$uac_i = ((Relationship, kid), \{(Adult, True), (FrontDoorLockToken, True)\})$$

This constraint implies that for any user u_x with an attribute value pair $(Relationship, kid)$ (in other words for any user u_x with $Relationship(u_x) = kid$), that user u_x cannot have the following set of attribute functions values pairs $\{(Adult, True), (FrontDoorLockToken, True)\}$. In other words, u_x cannot be assigned to: (1) the attribute $Adult$ with the value $True$ and (2) the attribute $FrontDoorLockToken$ with the value $True$.

Constraints on Session Attributes. These constraints enforce restrictions on session attributes. Here, an individual user may be assigned to different attributes simultaneously, and there is no conflict of interest when these attribute values are inherited independently by different sessions of the same user. However, there is a conflict of interest when inherited by the same session. In defining this type of constraint, we use the user attribute pairs set UAP similarly to how we use them in users' attributes' constraints as shown in Table 3. The difference is that users' attributes' constraints are enforced on the set of attributes available for each user. In contrast, sessions' attributes' constraints are enforced on the set of attributes that can be inherited by one session.

$U = \{alex, bob, anne, suzanne, john\}$, $USA = \{FamilyRole, FrontDoorLockToken\}$
$FamilyRole : U \rightarrow \{parent, kid, teenager\}$
$FamilyRole(alex) = FamilyRole(suzanne) = kid$, $FamilyRole(anne) = FamilyRole(john) = teenager$, $FamilyRole(bob) = parent$
$FrontDoorLockToken : U \rightarrow \{True, False\}$. This attribute is a dynamic attribute that is dynamically set by home owner.

$D = \{Oven, Fridge, FrontDoorLock, TV, PlayStation\}$
$OP = OP_{Oven} \cup OP_{Fridge} \cup OP_{FrontDoorLock} \cup OP_{PlayStation} \cup OP_{TV}$, where:
$OP_{Oven} = \{On_{oven}, \mathit{Off}_{oven}, Open_{oven}, Close_{oven}\}$, $OP_{Fridge} = \{Open_{fridge}, Close_{fridge}, CheckTemperature_{fridge}\}$,
$OP_{FrontDoorLock} = \{Lock_{FrontDoorLock}, Unlock_{FrontDoorLock}\}$, $OP_{PlayStation} = \{On_{PS}, \mathit{Off}_{PS}\}$, $OP_{TV} = \{On_{TV}, \mathit{Off}_{TV}, G_{TV}, PG_{TV}, R_{TV}\}$
$ops(Oven) = OP_{Oven}, ops(Fridge) = OP_{Fridge}, ops(FrontDoorLock) = OP_{FrontDoorLock}, ops(PlayStation) = OP_{PlayStation}, ops(TV) = OP_{TV}$
$DA = \{DangerousKitchenDevices, FrontDoorLockDevice, EntertainmentDevices, DeviceTemperature, UsingStatus, UsingUser\}$
$DangerousKitchenDevices : D \rightarrow \{True, False\}$
$DangerousKitchenDevices(Oven) = True$, $DangerousKitchenDevices(Fridge) = False$. All other values are undefined and evaluated to False
$FrontDoorLockDevice : D \rightarrow \{True, False\}$
$FrontDoorLockDevice(FrontDoorLock) = True$. All other values are undefined
$EntertainmentDevices : D \rightarrow \{True, False\}$
$EntertainmentDevices(TV) = EntertainmentDevices(PlayStation) = True$. All other values are undefined and evaluated to False
$DeviceTemperature : D \rightarrow \{x | x$ *is an oven temperature*$\}$
$DeviceTemperature(Oven)$ is dynamically set by sensors. All other values are undefined
$UsingStatus : D \rightarrow \{True, False\}$
$UsingStatus(TV)$ and $UsingStatus(PlayStation)$ are dynamically set by sensors. All other values are undefined
$UsingUser : D \rightarrow U$
$UsingUser(TV)$ and $UsingUser(PlayStation)$ are s dynamically set by sensors. All other values are undefined
$OPA = \{KidsFriendlyContent, DangerousKitchenOperation\}$
$KidsFriendlyContent : OP \rightarrow \{True, False\}$
$KidsFriendlyContent(G_{TV}) = KidsFriendlyContent(On_{TV}) = KidsFriendlyContent(\mathit{Off}_{TV}) = KidsFriendlyContent(On_{PS}) = KidsFriendlyContent(\mathit{Off}_{PS}) = True$
$KidsFriendlyContent(PG_{TV}) = KidsFriendlyContent(R_{TV}) = False$. All other values are undefined
$DangerousKitchenOperation : OP \rightarrow \{True, False\}$
$DangerousKitchenOperation(On_{Oven}) = DangerousKitchenOperation(Open_{Oven}) = True$
$DangerousKitchenOperation(\mathit{Off}_{Oven}) = DangerousKitchenOperation(Close_{Oven}) = DangerousKitchenOperation(Open_{Fridge}) = DangerousKitchenOperation(Close_{Fridge}) = DangerousKitchenOperation(CheckTemperature_{Fridge}) = False$. All other values are undefined

$ES = \{current\}$, $ESA = \{day, time, ParentInKitchen\}$, $day : ES \rightarrow \{S, M, T, W, Th, F, Sa\}$, $time : ES \rightarrow \{x | x$ is an hour of a day$\}$,
$ParentInKitchen : ES \rightarrow \{True, False\}$

$R = \{kid\}$, $roles(alex) = roles(suzanne) = \{kid\}$, $roles(anne) = roles(john) = roles(bob) = \emptyset$
$PRConstraints = \{(\{(Oven, On_{Oven}), (Oven, \mathit{Off}_{Oven}), (Fridge, Open_{Fridge}), (Fridge, Close_{Fridge})\}, \{kid\})\}$

$Authorization(s : S, op : OP, d : D, current : ES) \equiv$
$(parent \in FamilyRole(s)) \vee$
$(teenager \in FamilyRole(s) \wedge ParentInKitchen(current) \wedge DangerousKitchenDevices(d) \wedge DangerousKitchenOperation(op) \wedge Device\_Temperature(d) \leq 150°) \vee$
$(teenager \in FamilyRole(s) \wedge \neg DangerousKitchenOperation(op)) \vee$
$(teenager \in FamilyRole(s) \wedge FrontDoorLockDevice(d) \wedge FrontDoorLockToken(s)) \vee$
$(teenager \in FamilyRole(s) \wedge day(current) \in \{Sa, S\} \wedge 17{:}00 \leq time(current) \leq 23{:}59) \wedge EntertainmentDevices(d) \wedge (\neg UsingStatus(d) \vee UsingUser(d) = user(s)) \vee$
$(kid \in FamilyRole(s) \wedge day(current) \in \{Sa, S\} \wedge 17{:}00 \leq time(current) \leq 19{:}00) \wedge EntertainmentDevices(d) \wedge KidsFriendlyContent(op) \wedge (\neg UsingStatus(d) \vee UsingUser(d) = user(s))$

Fig. 4. HyBAC$_{AC}$ use case configuration.

HyBAC$_{AC}$ is an operational access control model. Managing and enforcing different types of constraints can be considered part of an administrative access control model, which is outside the scope of this manuscript.

Attributes Authorization Function. An attribute authorization function is a boolean function that is evaluated for each access decision. It is defined using the grammar of Table 4. For a specific session s_i, operation op_k, and device d_j the authorization function $Authorization(s_i, op_k, d_j, current)$ is evaluated by substituting the actual attribute values of $usa(s_i)$, $opa(op_k)$, $da(d_j)$, and $esa(current)$ for the corresponding symbolic placeholders and evaluating the resulting logical formula to be True or False. Any term that references an undefined attribute value is evaluated as False. Term refers to any atomic logical declarative sentence. An atomic sentence is a type of declarative sentence that is either true or false and cannot be broken down into other sentences [82].

Check Access Predicate. This predicate is evaluated for each access request. When a session s_i attempts operation op_k on device d_j in context of environment state $current$ the $CheckAccess(s_i, op_k, d_j, current)$ predicate evaluates to True or False. The check access predicate performs three checks, as shown in the following:

1) $op_k \in ops(d_j)$ Check if the requested operation is actually allowed on the target device.
2) $Authorization(s_i, op_k, d_j, current)$) Check if the authorization function evaluates to True. This indicates that the user is allowed to perform the requested operation on the target device according to the predefined configured policies.
3) $(\forall (P_x, R_y) \in PRConstraints)[((op_k, d_j) \notin P_x) \vee (roles(user(s_i))) \cap R_y = \phi]$ Check whether the user who created the requesting session is prohibited from accessing the requested permission by one of the permission role constraints. This statement basically ensures that for each permission role constraint (P_x, R_x) in the set of $PRConstraint$ one or both of the following conditions are satisfied:
   a) The requested permission (op_k, d_j) is not in the set of permission P_x.
   b) The intersection between the set of assigned roles to the user who created the requesting session $user(s_i)$ and the set of roles R_y is equal to ϕ.

5.1 Use Case Demonstration

This section illustrates how to configure HyBAC$_{AC}$ to achieve the same goals of the use case presented in Section 4.1. For this purpose, HyBAC$_{AC}$ will be configured as shown in Fig. 4. We have five users *bob*, *alex*, *suzanne*, *john*, and *anne*. We have two user/session attribute functions, $FamilyRole$ and $FrontDoorLockToken$, respectively specify the user relationship to the house's family and whether the

user is authorized to use the front door lock or not. We have five devices $Oven$, $Fridge$, $FrontDoorLock$, $PlayStation$ and TV. Each device has a set of operations allowed to be performed on it. We have six device attribute functions. The device attribute function $DangerouseKitchenDevices$ specifies whether the device is a dangerous kitchen device or not. The function $FrontDoorLockDevice$ determines if the device is a front door lock device or not. The device attribute function $EntertainmentDevices$ specifies if the device is considered an entertainment device or not. The attribute function $DeviceTemperature$ captures the temperature of the device. The device attribute function $UsingStatus$ determines whether the device is currently in use or not. Finally, the device attribute function $UsingUser$ specifies who is currently using the device. Different devices are assigned to different device attribute values. We have two operation attributes. The operation attribute function $KidsFriendlyContent$ specifies whether the operation is a kid friendly operation or not. Moreover, the operation attribute function $DangerouseKitchenOperation$ specifies whether the operation is considered a dangerous kitchen operation or not. Different operations are assigned to different operation attribute values. The authorization function set $Authorization(s : S, op : OP, d : D, current : ES)$ is a disjunction of six propositional statements. The first statement gives parents access to anything unconditionally. The second statement authorizes teenagers to use dangerous kitchen operations on dangerous kitchen devices only when one of the parents is in the kitchen, and the device temperature is below $150°$. The third statement gives teenagers access to nondangerous kitchen operations unconditionally. The fourth statement allows a teenager to use the front door lock device only if the parent grants him/her the front door lock token. The fifth statement gives teenagers access to entertainment devices during a specific time and if another user does not use the requested device. Finally, the sixth statement permits kids to use kids friendly operations on entertainment devices during a specific time and if they are currently not in use by another user.

6 Implementation

In this section, we provide a proof of concept implementation of HyBAC$_{RC}$ and HyBAC$_{AC}$ models. The purpose is to validate the applicability of these models using commercially available systems. We enforced the use cases presented in Sections 4.1 and 5.1 respectively using AWS (Amazon Web Services) IoT service [84]. We implemented the smart home IoT architecture shown in Fig. 5, which was first introduced by Geneiatakis et al. [83]. According to this architecture, the IoT devices are connected to a central hub and cannot be accessed by users or other devices directly. Simulations have been used to reflect real smart home devices. AWS Greengrass [85] was utilized to act as a smart hub and policy engine. Access is generally divided into two types: (a) By using the hub's connectivity services, users can interact with IoT devices locally, or (b) through cloud services that communicate with the smart hub to provide remote access to IoT devices via the Internet. We configured Greengrass's lambda function to receive the request, analyze it according to the configured logic and saved files, and trigger the desired actions on the corresponding device. In our enforcement, we only handled local communication.

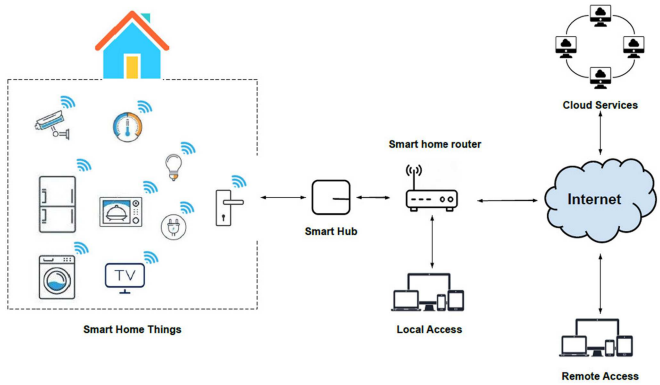

Fig. 5. Enforcement architecture (adapted from [83]).

We first created an AWS account, then configured and deployed Greengrass [85], which serves as a smart hub and a policy engine. The Greengrass SDK (Software Development Kit) extends cloud capabilities to the edge (in our case, the edge is the smart home). It enables devices to process data closer to the source of information and communicate securely on local networks. We deployed Greengrass on a dedicated virtual machine with one virtual CPU and 2 GB of RAM running ubuntu server 18.04.5 LTS. Then, we used AWS IoT device SDK for Python [86] provided by AWS on different virtual machines to simulate the users (devices used by users to access the smart things) and the smart devices (smart things that users want to perform different operations on them). Finally, we created a virtual object (digital shadow) for each physical device (users' devices or smart thing devices). Digital shadows are virtual counterparts of real physical IoT devices in the cloud. Shadows maintain the identity and last known state of the associated IoT device [87]. Each physical device and its corresponding shadow are cryptographically linked via digital certificates. MQTT protocol [88] is used by the devices and users to communicate to the AWS IoT service with TLS security [89]. MQTT standard is a machine-to-machine (M2M) lightweight publish/subscribe messaging protocol specially designed for constrained devices. Each shadow has a set of predefined MQTT topics/channels to interact with other IoT devices and applications.

HyBAC$_{RC}$ Enforcement. We created and configured three JSON files in the smart hub: (a) UsersRolesAssignment.jason. This defines the assignments of users to different roles. (b) ModelComponentConfiguration.json. This file defines and configures the rest of HyBAC$_{RC}$ components to express our use case. (c) UsersDevicesDynamicAttributes.json to capture different users' and devices' dynamic attributes. The process of updating the values of dynamic attributes in this JSON file is beyond the scope of this research.

HyBAC$_{AC}$ Enforcement. Here, we created and configured four JSON files in the smart hub as follows, UsersAttributes.json, DevicesAttributes.json, OperationAttributes.json, and EnvironmnetAttributes.json, to capture different users attributes, devices attributes, operations attributes, and environment attributes, respectively. The process of updating the values of different attributes in these JSON files is beyond the scope of this research.

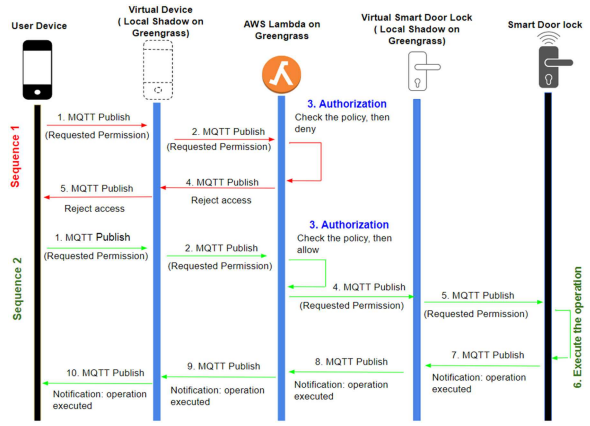

Fig. 6. The sequence of events in local communication.

The local communication sequence of actions is illustrated in Fig. 6. The sequence of events illustrated in red and tagged as sequence 1 depicts a denied request. On the other hand, The sequence of events illustrated in green and tagged as sequence 2 depicts an authorized request. For example, when the user tries to send permission requests to unlock the front door lock through his mobile phone while inside the house. First, a request is sent via MQTT protocol to the virtual object (or local shadow) corresponding to the user phone in Greengrass through the publish/subscribe relation between the user's phone and the local shadow. The local shadow gets notified of the request and sends it to the lambda function through MQTT publish/subscribe protocol. The lambda function analyzes the request according to the model implemented. In the case of HyBAC$_{RC}$, it analyzes the request according to the UsersRolesAssignment.json, ModelComponentConfiguration.json, and UsersDevicesDynamicAttributes.json files and decides whether to authorize the user to unlock the front door or not. On the other hand, in the case of HyBAC$_{AC}$, it analyzes the request according to the content of UsersAttributes.json, DevicesAttributes.json, OperationAttributes.json, and EnvironmnetAttributes.json files and makes the decision. If the request is denied, the lambda function publishes to the user's shadow update topic, and the local shadow gets notified and updates the user's phone that the permission was denied. In this case, the front door lock does not get an indication that a user attempted to access it. If the request is granted, the front door lock local shadow is notified through its update topic and updates the front door lock with the unlock command. After the front door lock is unlocked, it notifies its shadow by publishing to the shadow update topic. The front door lock local shadow then notifies the lambda function, which notifies the user phone's local shadow. Finally, the user phone's local shadow updates the user's phone that the TV was turned on successfully.

TABLE 5
One User Sending Requests to Multiple Devices

Users	Devices	$HyBAC_{RC}$ L.P.T	$HyBAC_{AC}$ L.P.T	N.R
1	1	1.8343 ms	1.2661 ms	10
1	3	1.7408 ms	1.3118 ms	30
1	5	1.76588 ms	1.3503 ms	50

TABLE 6
Multiple Concurrent Instances of One User Sending Request to One Device

Users	Devices	$HyBAC_{RC}$ L.P.T	$HyBAC_{AC}$ L.P.T	N.R
1	1	1.8343 ms	1.2661 ms	10
3	3	1.8385 ms	1.3803 ms	30
5	5	2.01128 ms	1.3247 ms	50

6.1 Performance Results

We executed multiple test scenarios to check the policy machine's response in each case. Furthermore, we analyzed the performance of our implementation. In particular, we measured the average lambda function processing time under different conditions. We implemented the configuration of the same use case given in Section 4.1, and Section 5.1 for HyBAC$_{RC}$ and HyBAC$_{AC}$ models respectively. We analyzed three different cases with three different loads of requests, each unique load of requests was executed ten times to measure the average lambda processing time as follows. (a) When one user is sending requests to multiple devices at the same time. Table 5 shows the measured average lambda function processing time in this test case for $HyBAC_{RC}$ and $HyBAC_{AC}$ implementation. The first row shows the average time when the parent, Bob, requests to lock the front door lock. The second row shows the average time when Bob requests to lock the front door lock, turn on the TV, and turn on the PlayStation simultaneously. The third row shows the average time when Bob requests to lock the front door lock, open the fridge, and turn on the oven, the TV, and the PlayStation simultaneously. All the requests were approved as they were supposed to according to our configured policies. (b) When multiple users are sending requests to multiple devices simultaneously (one user per device). Table 6 describes the measured average lambda function processing time in this test case for HyBAC$_{RC}$ and HyBAC$_{AC}$ implementation. The first, second, and third rows show the average time when the parent Bob requests to lock the front door lock, the average time when Bob requests to lock the front door lock, the kid Suzzane requests to turn on the oven, and the teenager John requests to open the fridge at the same time, the average time when the three access requests tested in the second row are carried again in addition to, the kid Alex requests to turn on the TV, and the teenager Anne requests to open the oven while one of the parents is in the kitchen and the oven temperature is 100°. The two systems responded correctly where all the requests were granted except for when the kid Suzzane was trying to turn on the oven since, according to our configuration, she is not allowed to, and when Alex was trying to turn on the TV since the testing was performed during a weekday, and according to our configuration kids are not allowed to access TV during weekdays. (c)Finally, Table 7 illustrates the average lambda function processing time when multiple users are sending requests to one device at the same time in HyBAC$_{RC}$ and HyBAC$_{AC}$ implementation. The first row illustrates the average time when Bob requested to unlock the front door lock. The second row shows the average time when Bob, Suzzane, and Alex simultaneously requested to unlock the

TABLE 7
Multiple Users Sending Requests to One Device

Users	Devices	$HyBAC_{RC}$ L.P.T	$HyBAC_{AC}$ L.P.T	N.R
1	1	1.8343 ms	1.2661 ms	10
3	1	1.73177 ms	1.2818 ms	30
5	1	1.8771 ms	1.2654 ms	50

L.P.T ≡ Lambda function processing time in milliseconds.
N.R ≡ Total number of requests (10 per unique request)

front door lock. Finally, the last row shows the average lambda processing time when Bob, Suzzane, Alex, John, and Anne all requested to unlock the front door lock at the same time. The two systems responded correctly, where all the requests were denied except when Bob requested to lock the front door lock.

6.1.1 Results Analysis

a. Average Processing Time

From the tables, we can notice that the two models are functional and applicable using commercially available technology. Furthermore, there is no specific pattern in the average lambda processing time when the number of requests increases in Tables 5, 6, and 7. In general, the processing time is short. The main reason behind that is that in our enforcement, we deployed the Greengrass, which serves as a smart hub on a dedicated virtual machine with one virtual CPU and 2 GB of RAM. This specification enables it to handle this number of requests without decreasing performance. However, this may not be the case if the number of requests increases drastically, which is not typical in a smart home IoT environment where we have a limited number of devices and users, unlike the case in other IoT domains.

b. $HyBAC_{AC}$ Average Processing Time is Always Less than $HyBAC_{RC}$ Average Processing Time

From the results, we can notice that the lambda average processing time in $HyBAC_{RC}$ is consistently higher than that in $HyBAC_{AC}$. The main reason is that the $HyBAC_{RC}$ CheckAccess predicate checking process is more complicated than the $HyBAC_{AC}$ CheckAccess predicate checking process.

In $HyBAC_{RC}$, if a session s_i attempts to perform an operation op_k on a device d_j when the subset of environment conditions EC_l are active, the CheckAccess predicate checks the following:

1) The requirements of role membership and role activation specified by the underlying role-based model are satisfied. In other words, ensure that the requesting user currently active roles enable him to access at least one of the device roles that the requested permission is assigned to under the current active environment roles. This check will evaluate to true if and only if there is a role pair rp_m and a device role dr_n assigned to each other through the relation $RPDRA$ such that the following conditions are true:
   a) The device role dr_n is assigned to the permission (d_j, op_k) through the relationship $PDRA$.
   b) The role part of the role pair rp_m, which is $rp_m.r$ is one of the currently active roles of the session s_i.
   c) Each environment role in the set of environment roles part of the role pair rp_m, which we refer to by $rp_m.ER$ is active because it is activated by a subset of the currently active environment conditions EC_l.
2) The authorization function, which checks the user's and device's dynamic attributes, is satisfied. In other words, the user's current dynamic attributes enable him to access the requested device with the current dynamic device attribute.

On the other hand, in $HyBAC_{AC}$ the CheckAccess predicate only checks the following:

1) The authorization function ensures that the requesting user is allowed to access the requested operations on the requested device according to the current attribute values.
2) The set of permission role constraints $PRConstraints$, to ensure that the requested user is not prohibited from accessing the requested permission by one of the permission role constraints.

7 Theoretical Comparison

It is crucial to select an access control model appropriate to an organization's structure, requirements, and specifications to achieve optimal results and minimize access risks and threats. This section compares four access control models designed to meet smart home IoT challenges. We compare the hybrid models this paper developed, $HyBAC_{RC}$, and $HyBAC_{AC}$ with EGRBAC and HABAC. We evaluate these models against access control criteria adapted from [16] toward this goal. These criteria are grouped into two categories: (a) basic criteria and (b) quality criteria.

7.1 Basic Criteria

This type of criteria consists of six elements. Here we discuss each criterion.

a. Least Privilege Principle. All four models support this principle. Users who belong to a variety of roles (in EGRBAC, or $HyBAC_{RC}$) or have different attributes corresponding to their roles in the house (in HABAC, and $HyBAC_{AC}$) can utilize any subset of them that will allow tasks to be accomplished.

b. Attributed Based Specifications. We have two types of attributes, as explained in Section 2, static and dynamic. The four models support environment attributes. Moreover, they all support static users, devices, and operations attributes (conditions). Furthermore, while HABAC, $HyBAC_{RC}$, and $HyBAC_{AC}$ models all support dynamic attributes, EGRBAC does not.

c. Constraints. These are invariants that must be maintained. We have three types of constraints, static separation of duty, dynamic separation of duty, and permission-role constraints, as explained in Sections 4 and 5. From Table 8, we notice that all four models support the three types of constraints except for HABAC, which does not support permission-role constraints [10].

d. Authentication. All four models support positive (close) authentication. The closed policy permits access when there is a positive authorization for such access and denies it otherwise.

e. Access Administration. Here, we compare the four models based on two administrative tasks. Users and devices provisioning and configuration effort. In general, the user or device provisioning is easier in RBAC-based models (including EGRBAC and HyBAC$_{RC}$) than in ABAC-based models (including HABAC and HyBAC$_{AC}$). In RBAC models, users' and devices' provisioning simply requires the administrator (the homeowner) to assign different roles to the newly created users or devices. On the other hand, in ABAC-based models, the administrator needs to configure different attribute values for newly provisioned users or devices. Regarding configuration effort, from Table 8, we can notice that HyBAC$_{RC}$ requires more configuration effort than the other three models.

f. Access Review. In RBAC based model (including EGRBAC), to determine the set of permissions available for each user, we just look into the set of roles assigned to it. Similarly, in HyBAC$_{RC}$, by looking into the set of roles assigned to each user, we can determine the maximum set of permissions available for each user. However, defining the set of permissions available for each user could be more complicated in ABAC-based models (such as HABAC and HyBAC$_{AC}$) [14].

g. Administrative Policies. To determine how administrative privileges are organized in any model, an access control administration model is required. However, in the four models, it is assumed that the house owner is the one who is responsible for granting or revoking permissions. Hence, we can say that they all support centralized administrative policies.

7.2 Quality Criteria

As explained in the following, we have three crucial factors.

a. Expressiveness and Meaningfulness. Three features are required for an AC model to be expressive. First, a formal definition is needed to specify the intended behavior in detail. The second requirement is that it must be sufficiently meaningful and comprehensive enough to support a variety of constraints. Lastly, the model should include both static and dynamic attributes. All four models that we are comparing are formally defined. Furthermore, they all support different constraints except HABAC since it does not support permission-role constraints. All four models capture different types of attributes except for EGRBAC, which does not capture the user's and device's dynamic attributes.

b. Flexibility To consider a specific AC model as a flexible model, several factors need to be assessed. Here we identify three of them. First, the model should be flexible enough to meet smart home IoT requirements. Second, the model should support delegation, which means the capability of a user to delegate his privileges to any other user partially or totally. Lastly, the flexibility of provisioning new components. Regarding the first factor and using the criteria proposed in [15], for an access control model to address smart home IoT requirements, it should be dynamic, fine-grained, and suitable for constrained smart home devices. In addition, the model should be developed specifically for smart home IoT or interpreted in light of appropriate smart home use cases. The model should be implemented to be credible using commercially available technology. Finally, the model should be formalized so that its intended behavior can be specified precisely. As discussed in [15] EGRBAC meet this criteria. However, as we discussed earlier, EGRBAC does not support dynamic attributes. On the other hand, HABAC, HyBAC$_{RC}$, and HyBAC$_{AC}$ are dynamic, fine-grained, suitable for the constrained home environment, designed specifically for smart home IoT, illustrated with a use case demonstration, have proof of concept implementation, and they are formally defined. Hence, they meet the criteria proposed in [15]. Regarding delegation support, this cannot be entirely determined without an access control administration model. But generally speaking, it has been shown before that RBAC and ABAC based models can perform delegation. All four models are capable of provisioning new users and devices. However, as we explained in Section 7.1, it is easier to provision new users and devices in EGRBAC and HyBAC$_{RC}$ than in HABAC and HyBAC$_{AC}$.

c. Efficiency Level and Scalability An access control model is required to answer two main questions on efficiency and scalability. Is the model can easily be expanded? If not, it will be unreliable in the real world. Moreover, is the model expansion affect its efficiency negatively? The answers to these questions can only be obtained through a more detailed study. However, generally speaking, we are dealing with a relatively small number of users and devices in smart home IoT. Moreover, ABAC and RBAC based models have proved their scalability since they have been widely adopted in different organizations of enormous sizes.

8 Discussion

As discussed in Section 2 and as many researchers recently described [9], [10], [11], [12], [13], [14], the need arises for hybrid models that combine ABAC and RBAC based models advantages while eliminating their disadvantages to meet access control requirements in smart IoT systems.

In this paper, we proposed HyBAC$_{RC}$. It is a role-centric hybrid model that governs users to devices access control. It is developed based on the EGRBAC model [15], which is a role-based access control model developed for smart home IoT environment. As in HABAC and unlike EGRBAC, HyBAC$_{RC}$ can handle dynamic attributes. As discussed in Section 2, handling dynamic users' or devices' attributes in EGRBAC may result in role explosion. For example, consider a use case where the homeowner wants to grant teenagers use of dangerous permissions in kitchen devices (oven and gas stove), such as opening the oven, turning on the oven, and turning on the gas stove but only when the device temperature is below a threshold, say 150°. This use case can easily be handled in ABAC models such as HABAC [10] by defining a dynamic device attribute $device\_temperature$, which measures the device temperature, and then configuring an access policy that authorizes teenagers to access dangerous permissions in kitchen devices only if the attribute $device\_temperature$ is below 150°. In EGRBAC, on the other hand, the component responsible for categorizing permissions of different devices based on different characteristics is the *device roles (DR)*. Different users get access to different permissions by assigning the users' roles to the device roles of those permissions. Hence, we could try to capture this use case in EGRBAC by defining two device roles: (a) $High\_Temperature\_Permissions$ and

TABLE 8
Evaluating Smart Home IoT Access Control Models Against Basic Criteria

Criteria	EGRBAC	HABAC	HyBAC$_{RC}$	HyBAC$_{AC}$
a. Least privilege principle	Yes	yes	yes	yes
b. Attributed based specifications				
a. Static	Yes	Yes	yes	Yes
b. Dynamic	No	yes	yes	yes
c. Constraints				
a. Static separation of duty	Yes	Yes	yes	Yes
b. Dynamic separation of duty	Yes	yes	yes	yes
c. P-R constraints	Yes	No	yes	yes
d. Authentication	Positive(Close)	Positive(Close)	Positive(Close)	Positive(Close)
e. Access administration				
a. Users and Devices provisioning	Easy	Complicated	Easy	Complicated
c. Configuration effort	1- Define and set initial users, devices, and operations static characteristics (user roles, and device roles) 2- Define environment conditions, environment roles, and environment activations 3- Setting up initial role structure and assignments	1- Define and set initial users, devices, and operations static and dynamic characteristics (user roles, and device roles) 2- Define environment states, and environment state attributes 3- Specify access policies	1- Define and set initial users, devices, and operations static and dynamic characteristics (attributes) 2- Define environment conditions, environment roles, and environment activations 3- Setting up initial role structure and assignments 4- Specify access policies	1- Define and set initial users, devices, and operations static and dynamic characteristics (attributes) 2- Define environment states, and environment state attributes 3- Specify access policies
f. Access review	Easy	Complicated	Easy	Complicated
g. Administrative policies	Centralized	Centralized	Centralized	Centralized

assigning to it the permissions that can be performed when the device temperature is high (which are: close the oven, turn off the oven, and turn off the gas stove only). (b) $Low\_Temperature\_Permissions$ and assign to it the permissions that can be performed when the device temperature is low (which is basically all oven and gas stove permissions). However, when permissions are assigned to a specific device role, they remain associated with that role until an administration change is made. Without administrative action, it is impossible to activate and deactivate permissions assignments to different device roles dynamically. Suppose that the gas stove temperature is less than 150° while the oven temperature is more than 150°. In this case, teenagers should get access to all gas stove permissions assigned to the device role $Low\_Temperature\_Permissions$ while allowed to access only those permissions assigned to $High\_Temperature\_Permissions$ in the oven while prevented from accessing oven permissions assigned to the device role $Low\_Temperature\_Permissions$. However, there is no mechanism to deactivate the assignment of the oven permissions to the device role $Low\_Temperature\ \_Permissions$ when the oven temperature is above 150° and activate it otherwise. Without such a mechanism, teenagers will access all oven permissions regardless of the oven's temperature. Another way to solve this problem is by creating two device roles for each device, one for high temperature permissions and another for low temperature permissions. In our use case, this would require a total of four device roles ($GasStove\_\ Low\_\ Temperature\_\ Permissions$, $GasStove\_\ High\_\ Temperature\_\ Permissions$, $Oven\_\ Low\_\ Temperature\_\ Permissions$, and $Oven\_\ High\_\ Temperature\_\ Permissions$). Still, there is no mechanism to activate and deactivate different device roles dynamically according to the different values of the device temperature. Suppose the gas stove temperature is low while the oven temperature is high. The model has no mechanism to activate the device roles $GasStove\_\ Low\_\ Temperature\_\ Permissions$ and $Oven\_\ High\_\ Temperature\_\ Permissions$ while deactivating the device roles $GasStove\_\ High\_\ Temperature\_\ Permissions$ and $Oven\_\ Low\_\ Temperature\ \_Permissions$. Without such a mechanism, teenagers will access all gas stove and oven permissions regardless of gas stove and oven temperatures. Moreover, increasing numbers of devices and dynamic attributes will lead to role explosion. The problem is further aggravated in other use cases, which may involve more devices and different dynamic device attributes with different values. For instance, we may want to grant access to different permissions based on devices' different temperature values, not only on whether the device temperature is above or below a specific value. In that case, the role explosion problem will worsen since we may need at least one device role for each device temperature value. See supplemental material for more information, available online.

On the other hand, HyBAC$_{RC}$ encapsulates relatively static attributes of access decisions in user roles and device roles while utilizing the concept of user's and device's dynamic attributes to capture rapidly changing attributes to constrain the permissions available for each user further. Therefore, the users' and devices' role sets determine the maximum set of available permissions, supporting the principle of least privilege and allowing easy review of user permissions. On the other hand, the users' and devices' dynamic attributes determine a flexible set of permissions for each user within the scope of his/her assigned roles. Furthermore, HyBAC$_{RC}$ can capture the environment contextual information through the component environment roles. It retains advantages of EGRBAC (such as ease of user provisioning, least privilege principle, and the ability to quickly determine and control the maximum permissions available to each user) while capturing different authorizations for

every possible user, environment, operation, and device dynamic conditions without risking role explosion.

Moreover, in this paper, we introduced HyBAC$_{AC}$. It is an attribute-centric hybrid model that governs user to device access control. It is developed based on the HABAC model [10], which is an attribute-based access control model developed for smart home IoT environment. HyBAC$_{AC}$ captures different users', environment's, operations', and devices' static and dynamic characteristics. Unlike HABAC, HyBAC$_{AC}$ supports the permission-role constraints by introducing the notion of roles (aka anti-roles). However, it enforces this type of constraint during execution time. On the other hand, EGRBAC and HyBAC$_{RC}$ enforce permission-role constraints during configuration time. HyBAC$_{RC}$ and HyBAC$_{AC}$ have similar expressiveness power; they both capture static and dynamic attributes. Moreover, both can express static separation of duty, dynamic separation of duty, and permission-role constraints. Both models are formally defined and illustrated through use case scenarios.

In Section 6, we provide a proof of concept implementation of HyBAC$_{RC}$ and HyBAC$_{AC}$ models. The purpose was to verify the applicability of these models using commercially available systems. Overall, both models are functional and applicable. However, as discussed in the same section, while the average lambda processing time in both models is generally law, the lambda average processing time in HyBAC$_{RC}$ is consistently higher than that in HyBAC$_{AC}$. The main reason is that the HyBAC$_{RC}$ CheckAccess predicate checking process is more complicated than that in HyBAC$_{AC}$.

Finally, in Section 7, we compare the two hybrid models developed in this paper, HyBAC$_{RC}$ and HyBAC$_{AC}$ with EGRBAC (an RBAC model specifically designed to meet smart home IoT challenges) and HABAC (an ABAC model specifically designed to meet smart home IoT challenges). From our comparison, we can notice that HyBAC$_{RC}$ and HyBAC$_{AC}$ are more expressive than EGRBAC and HABAC. Furthermore, we can notice that while HyBAC$_{AC}$ requires less configuration effort than HyBAC$_{RC}$, however, provisioning new users and devices and access review tasks are more straightforward in HyBAC$_{RC}$.

We acknowledge that practical smart homes will have different and more complex scenarios, and a detailed performance evaluation is ultimately necessary for simulating a large number of smart things. Nevertheless, our proof of concept implementation on AWS demonstrates the practical utility and use of fine-grained security policies within the context of smart homes IoT without the need to capture a large set of scenarios from the real world since a scaled setting will not affect the evaluation of security policies. However, we are considering extending this work to include a more detailed performance analysis. Since smart home residents are usually constrained and not willing to deal with complicated systems, one of the important aspects that need to be considered in smart home access control models is usability. A user or homeowners' related study needs to be performed to define a common preferable default setting policies. Additionally, a simple, usable, and expressive user interface need to be developed to mediate users' interaction with the access control system. These are future directions to consider.

9 Conclusion

In this paper, we proposed two hybrid models that govern users to devices access control in smart home IoT HyBAC$_{RC}$ and HyBAC$_{AC}$. HyBAC$_{RC}$ is a role-centric hybrid model that is built on top of the EGRBAC model (an RBAC model developed specifically to meet smart home IoT requirements). It encapsulates relatively static attributes of access decisions in user roles and device roles. It utilizes the concept of user's and device's dynamic attributes to capture rapidly changing attributes to constrain the permissions available for each user further. HyBAC$_{AC}$ is an attribute-centric model based on the HABAC model (an ABAC model developed specifically to meet smart home IoT requirements). Each model is formally defined and illustrated with a use case scenario. Moreover, each model is demonstrated with a proof of concept implementation in Amazon web services (AWS). Finally, we conducted a theoretical comparison between HyBAC$_{RC}$, HyBAC$_{AC}$, EGRBAC, and HABAC. We showed that HyBAC$_{RC}$ and HyBAC$_{AC}$ meet smart home IoT access control requirements and have similar expressive power. However, choosing between them will be a trade-off between considerable front role structuring effort for ease of administration and access review in the HyBAC$_{RC}$, on the one hand, and between easy setup effort but more complicated administration and access review tasks in the HyBAC$_{AC}$ model on the other hand.

References

[1] F. K. Aldrich, "Smart homes: Past, present and future," in *Inside the Smart Home*, Berlin, Germany: Springer, 2003.

[2] W. He et al., "Rethinking access control and authentication for the home Internet of Things (IoT)," in *Proc. 27th USENIX Conf. Secur.*, 2018, pp. 255–272.

[3] A. Tilley, How a few words to Apple's Siri unlocked a man's front door, 2016. [Online]. Available: http://www. forbes. com/sites/aarontilley/2016/09/21/apple-homekit-siri-security

[4] K. Hill, Baby monitor hack could happen to 40,000 other foscam users, 2013. [Online]. Available: https://www.forbes.com/sites/kashmirhill/2013/08/27/baby-monitor-hack-could-happen-to-40000-other-foscam-users/613ec55458b5

[5] B. Bezawada et al., "Securing home IoT environments with attribute-based access control," in *Proc. 3rd ACM Workshop Attribute-Based Access Control*, 2018, pp. 43–53.

[6] S. Bhatt et al., "Access control model for AWS Internet of Things," in *Proc. Int. Conf. Netw. Syst. Secur.*, 2017, pp. 721–736.

[7] N. Ye et al., "An efficient authentication and access control scheme for perception layer of Internet of Things," *Appl. Math. Inf. Sci.*, vol. 8, no. 4, pp. 1617–1624, 2014.

[8] S. Kaiwen et al., "Attribute-role-based hybrid access control in the Internet of Things," in *Proc. Asia-Pacific Web Conf.*, 2014, pp. 333–343.

[9] M. U. Aftab et al., "A hybrid access control model with dynamic COI for secure localization of satellite and IoT-based vehicles," *IEEE Access*, vol. 8, pp. 24 196–24 208, 2020.

[10] S. Ameer et al., "The HABAC model for smart home IoT and comparison to EGRBAC," in *Proc. ACM Workshop Secure Trustworthy Cyber-Phys. Syst.*, 2021, pp. 39–48.

[11] A. Chatterjee et al., "Dynamic role-based access control for decentralized applications," in *Proc. Int. Conf. Blockchain*, 2020, pp. 185–197.

[12] D. Gupta, S. Bhatt, M. Gupta, O. Kayode and A. S. Tosun, "Access control model for google cloud IoT," in *Proc. IEEE 6th Int. Conf. Big Data Secur. Cloud, IEEE Int. Conf. High Perform. Smart Comput., IEEE Int. Conf. Intell. Data Secur.*, 2020, pp. 198–208.

[13] A. Thakare, E. Lee, A. Kumar, V. B. Nikam and Y. -G. Kim, "PARBAC: Priority-attribute-based RBAC model for Azure IoT cloud," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2890–2900, Apr. 2020.

[14] D. R. Kuhn, E. J. Coyne and T. R. Weil, "Adding attributes to role-based access control," *Computer*, vol. 43, no. 6, pp. 79–81, Jun. 2010.

[15] S. Ameer, J. Benson and R. Sandhu, "The EGRBAC model for smart home IoT," in *Proc. IEEE 21st Int. Conf. Inf. Reuse Integration Data Sci.*, 2020, pp. 457–462.

[16] S. M. Hasani et al., "Criteria specifications for the comparison and evaluation of access control models," *Int. J. Comput. Netw. Inf. Secur.*, vol. 5, no. 5, pp. 19–29, 2013.

[17] M. Alramadhan et al., "An overview of access control mechanisms for Internet of Things," in *Proc. IEEE 26th Int. Conf. Comput. Commun. Netw.*, 2017, pp. 1–6.

[18] J. Jindou, Q. Xiaofeng and C. Cheng, "Access control method for web of things based on role and SNS," in *Proc. IEEE 12th Int. Conf. Comput. Inf. Technol.*, 2012, pp. 316–321.

[19] S. Ameer et al., "An attribute-based approach toward a secured smart-home IoT access control and a comparison with a role-based approach," *Information*, vol. 13, no. 2, 2022, Art. no. 60.

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

[21] X. Jin et al., "A unified attribute-based access control model covering DAC, MAC and RBAC," in *Proc. IFIP Annu. Conf. Data App. Sec.*, 2012, pp. 41–55.

[22] O. Arias, J. Wurm, K. Hoang and Y. Jin, "Privacy and security in Internet of Things and wearable devices," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 1, no. 2, pp. 99–109, Second Quarter 2015.

[23] B. Ur et al., "The current state of access control for smart devices in homes," in *Proc. Workshop Home Usable Privacy Secur.*, 2013, pp. 209–218.

[24] P. M. Chanal et al., "Security and privacy in IoT: A survey," *Wirel. Pers. Commun.*, vol. 115, pp. 1667–1693, 2020.

[25] T. Denning et al., "Computer security and the modern home," *Commun. ACM*, vol. 56, no. 1, pp. 94–103, 2013.

[26] A. Cui et al., "A quantitative analysis of the insecurity of embedded network devices: Results of a wide-area scan," in *Proc. 26th Annu. Comput. Secur. Appl. Conf.*, 2010, pp. 97–106.

[27] J. Granjal et al., "Security for the Internet of Things: A survey of existing protocols and open research issues," *IEEE Commun. Surv. Tuts.*, vol. 17, no. 3, pp. 1294–1312, Third Quarter 2015.

[28] A. Jain et al., "Security challenges and solutions of IoT ecosystem," in *Proc. Inf. Commun. Technol. Sustain. Develop.*, 2020, pp. 259–270.

[29] T. Oluwafemi et al., "Experimental security analyses of non-networked compact fluorescent lamps: A case study of home automation security," in *Proc. Workshop Learn. Authoritative Secur. Exp. Results*, 2013, pp. 13–24.

[30] M. M. Ogonji et al., "A survey on privacy and security of Internet of Things," *Comput. Sci. Rev.*, vol. 38, 2020, Art. no. 100312.

[31] A. R. Sfar et al., "A roadmap for security challenges in the Internet of Things," *Digit. Commun. Netw.*, vol. 4, no. 2, pp. 118–137, 2018.

[32] M. Tabassum et al., "Smart home beyond the home: A case for community-based access control," in *Proc. Conf. Hum. Factors Comput. Syst.*, 2020, pp. 1–2.

[33] E. Fernandes, J. Jung and A. Prakash, "Security analysis of emerging smart home applications," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 636–654.

[34] G. Ho et al., "Smart locks: Lessons for securing commodity Internet of Things devices," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2016, pp. 461–472.

[35] E. Fernandes et al., "Flowfence: Practical data protection for emerging IoT application frameworks," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 531–548.

[36] P. Morgner et al., "All your bulbs are belong to us: Investigating the current state of security in connected lighting systems," 2016, *arXiv:1608.03732*.

[37] M. Gupta et al., "Authorization framework for secure cloud assisted connected cars and vehicular Internet of Things," in *Proc. 23nd ACM Symp. Access Control Models Technol.*, 2018, pp. 193–204.

[38] A. Ouaddah et al., "Access control in the internet of things: Big challenges and new opportunities," *Comput. Netw.*, vol. 112, pp. 237–262, 2017.

[39] Y. Zhang et al., "Access control in internet of things: A survey," 2016, *arXiv:1610.01065*.

[40] S. Ravidas et al., "Access control in internet-of-things: A survey," *J. Netw. Comput. Appl.*, vol. 144, pp. 79–101, 2019.

[41] J. Qiu et al., "A survey on access control in the age of Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4682–4696, Jun. 2020.

[42] E. Bertin et al., "Access control in the Internet of Things: A survey of existing approaches and open research questions," *Ann. Telecommun.*, vol. 74, no. 7, pp. 375–388, 2019.

[43] D. F. Ferraiolo et al., "Proposed NIST standard for role-based access control," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 224–274, 2001.

[44] G. Zhang and Jiazheng Tian, "An extended role based access control model for the Internet of Things," in *Int. Conf. Inf., Netw. Automat.*, 2010, pp. V1–319–V1-323.

[45] E. Barka et al., "Securing the web of things with role-based access control," in *Proc. Int. Conf. Codes, Cryptol., Inf. Secur.*, 2015, pp. 14–26.

[46] P. Spiess et al., "SOA-based integration of the Internet of Things in enterprise services," in *Proc. IEEE Int. Conf. Web Serv.*, 2009, pp. 968–975.

[47] L. M. S. De Souza et al., "SOCRADES: A web service based shop floor integration infrastructure," in *Proc. Internet Things*, 2008, pp. 50–67.

[48] J. Liu, Y. Xiao and C. L. P. Chen, "Authentication and access control in the Internet of Things," in *Proc. IEEE 32nd Int. Conf. Distrib. Comput. Syst. Workshops*, 2012, pp. 588–592.

[49] B. Ndibanje et al., "Security analysis and improvements of authentication and access control in the Internet of Things," *Sensors*, vol. 14, no. 8, pp. 14786–14805, 2014.

[50] S. Bandara, T. Yashiro, N. Koshizuka and K. Sakamura, "Access control framework for API-enabled devices in smart buildings," in *Proc. 22nd Asia-Pacific Conf. Commun.*, 2016, pp. 210–217.

[51] V. C. Hu, D. R. Kuhn, D. F. Ferraiolo and J. Voas, "Attribute-based access control," *IEEE Comput.*, vol. 48, no. 2, pp. 85–88, Feb. 2015.

[52] A. Mutsvangwa, B. Nleya and B. Nleya, "Secured access control architecture consideration for smart grids," in *Proc. IEEE PES PowerAfrica*, 2016, pp. 228–233.

[53] Y. Xie et al., "Three-layers secure access control for cloud-based smart grids," in *IEEE 82nd Veh. Technol. Conf.*, 2015, pp. 1–5.

[54] R. Zhang et al., "ABSAC: Attribute-based access control model supporting anonymous access for smart cities," *Secur. Commun. Netw.*, vol. 2021, pp. 1–11, 2021.

[55] M. Gupta et al., "Dynamic groups and attribute-based access control for next-generation smart cars," in *Proc. 9th ACM Conf. Data Appl. Secur. Privacy*, 2019, pp. 61–72.

[56] S. Bhatt et al., "ABAC-CC: Attribute-based access control and communication control for Internet of Things," in *Proc. 25th ACM Symp. Access Control Models Technol.*, 2020, pp. 203–212.

[57] E. V. H.J. Dennis, "Programming semantics for multiprogrammed computations," in *Commun. ACM*, vol. 9, no. 3, pp. 143–155, 1966.

[58] J. Park et al., "Towards usage control models: Beyond traditional access control," in *Proc. 7th ACM Symp. Access Control Models Technol.*, 2002, pp. 57–64.

[59] J. Park, "Usage control: A unified framework for next generation access control," Ph.D. dissertation, Dep. Inf. Softw. Eng., George Mason Univ., 2003.

[60] X. Zhang et al., "Formal model and policy specification of usage control," *ACM Trans. Inf. Syst. Secur.*, 2005, pp. 351–387.

[61] Z. Guoping et al., "The research of access control based on UCON in the Internet of Things," *J. Softw.*, vol. 6, no. 4, pp. 724–731, 2011.

[62] A. La Marra, F. Martinelli, P. Mori and A. Saracin, "Implementing usage control in internet of things: A smart home use case," in *Proc. IEEE Trustcom/BigDataSE/ICESS*, 2017, pp. 1056–1063.

[63] F. Martinelli et al., "Too long, did not enforce: A qualitative hierarchical risk-aware data usage control model for complex policies in distributed environments," in *Proc. 4th ACM Workshop Cyber-Phys. Syst. Secur.*, 2018, pp. 27–37.

[64] A. Ouaddah et al., "Towards a novel privacy-preserving access control model based on blockchain technology in IoT," in *Proc. Europe MENA Coop. Adv. Inf. Comm. Technol*, 2017, pp. 523–533.

[65] O. Novo, "Blockchain meets IoT: An architecture for scalable access management in IoT," *IEEE IoT J.*, vol. 5, no. 2, pp. 1184–1195, Apr. 2018.

[66] A. Dorri, S. S. Kanhere, R. Jurdak and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, 2017, pp. 618–623.

[67] S. Malani, J. Srinivas, A. K. Das, K. Srinathan and M. Jo, "Certificate-based anonymous device access control scheme for IoT environment," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9762–9773, Dec. 2019.

[68] M. J. Covington et al., "Generalized role-based access control for securing future applications," Georgia Tech, Tech. Rep., 2000. Accessed: Oct. 31, 2022.

[69] M. J. Covington et al., "Securing context-aware applications using environment roles," in *Proc. 6th ACM Symp. Access Control Models Technol.*, 2001, pp. 10–20.

[70] H. Yan et al., "IoT-FBAC: Function-based access control scheme using identity-based encryption in IoT," *Future Gener. Comput. Syst.*, vol. 95, pp. 344–353, 2019.
[71] M. Alshahrani et al., "Secure mutual authentication and automated access control for iot smart home using cumulative keyed-hash chain," *J. Inf. Secur. Appl.*, vol. 45, pp. 156–175, 2019.
[72] A. K. Sikder et al., "Multi-user multi-device-aware access control system for smart home," 2019, *arXiv:1911.10186*.
[73] G. Goyal et al., "Securing smart home iot systems with attribute-based access control," in *Proc. ACM Workshop Secure Trustworthy Cyber-Phys. Syst.*, 2022, pp. 37–46.
[74] H. B. Attia et al., "A new hybrid access control model for security policies in multimodal applications environments," *J. Univ. Comput. Sci*, vol. 24, pp. 392–416, 2018.
[75] V. Varadharajan, A. Amid and S. Rai, "Policy based role centric attribute based access control model policy RC-ABAC," in *Proc. IEEE Int. Conf. Comput. Netw. Commun.*, 2015, pp. 427–432.
[76] S. Pal et al., "Protocol-based and hybrid access control for the iot: Approaches and research opportunities," *Sensors*, vol. 21, no. 20, 2021, Art. no. 6832.
[77] J. Park et al., "Activity control design principles: Next generation access control for smart and collaborative systems," *IEEE Access*, vol. 9, pp. 151004–151022, 2021.
[78] T. Mawla et al., "Bluesky: Activity control: A vision for" active" security models for smart collaborative systems," in *Proc. 27th ACM Symp. Access Control Models Technol.*, 2022, pp. 207–216.
[79] M. Gupta et al., "Towards activity-centric access control for smart collaborative ecosystems," in *Proc. 26th ACM Symp. Access Control Models Technol.*, 2021, pp. 155–164.
[80] S. Ameer et al., "Bluesky: Towards convergence of zero trust principles and score-based authorization for iot enabled smart systems," in *Proc. 27th ACM Symp. Access Control Models Technol.*, 2022, pp. 235–244.
[81] R. Sandhu et al., "The nist model for role-based access control: Towards a unified standard," in *ACM Workshop on Role-Based Access Control*, 2000, pp. 47–63.
[82] Atomic sentence. Accessed: Oct. 31, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Atomic_sentence
[83] D. Geneiatakis et al., "Security and privacy issues for an IoT based smart home," in *Proc. 40th USENIX Conf. Usable Privacy Secur.*, 2017, pp. 65–80.
[84] AWS-IoT. Accessed: Oct. 31, 2022. [Online]. Available: https://aws.amazon.com/iot/
[85] AWS IoT greengrass. Accessed: Oct. 31, 2022. [Online]. Available: https://docs.aws.amazon.com/greengrass/latest/developerguide/what-is-gg.html
[86] AWS IoT device SDK for python. Accessed: Oct. 31, 2022. [Online]. Available: https://docs.aws.amazon.com/greengrass/latest/developerguide/IoT-SDK.html
[87] AWS-shadow. Accessed: Oct. 31, 2022. [Online]. Available: https://docs.aws.amazon.com/iot/latest/developerguide/iot-device-shadows.html
[88] MQTT.fx - a JavaFX based MQTT client. Accessed: Oct. 31, 2022. [Online]. Available: https://softblade.de/en/welcome/
[89] The transport layer security (TLS) protocol. Accessed: Oct. 31, 2022. [Online]. Available: https://tools.ietf.org/html/rfc5246

Safwa Ameer received the PhD degree in computer science from the University of Texas at San Antonio. She is currently a PostDoc researcher with the Institute for Cyber Security , University of Texas at San Antonio. Her main areas of interest include security and privacy in cyberspace.

James Benson received the master's degree in electrical engineering focusing on computer science from the University of Texas at San Antonio, and the master's degree in physics from Clarkson University. He is a technology research analyst with the Institute for Cyber Security, University of Texas at San Antonio.

Ravi Sandhu (Fellow, IEEE) is the founding executive director with the Institute for Cyber Security, University of Texas at San Antonio and lead PI of its NSF Center for Security and Privacy Enhanced Cloud Computing, where he holds the lutcher brown endowed chair in Cyber Security. He is a Fellow of the ACM, AAAS, and the National Academy of Inventors. He is an inventor on 31 patents.