The HABAC Model for Smart Home IoT and Comparison to EGRBAC

Safwa Ameer and Ravi Sandhu
Institute for Cyber Security and NSF Center for Security and Privacy Enhanced Cloud Computing
Department of Computer Science, University of Texas at San Antonio
San Antonio, Texas, USA
Safwa.Ameer@gmail.com,ravi.sandhu@utsa.edu

Abstract

In the near future IoT will be part of every home turning our houses into smart houses, in which we have multiple users with complex social relationships between them using the same smart devices. This requires sophisticated access control specification and enforcement models. Recently, several access control models have been developed or adapted for IoT in general, with a few specifically designed for the smart home IoT domain. The majority of these models are built on role-based access control (RBAC) or attribute-based access control (ABAC) models which have had considerable traction in traditional non-IoT domains. In this paper, we introduce the smart home IoT attribute-based access control model (HABAC). HABAC is a dynamic and fine-grained model that is developed specifically to meet smart home IoT challenges. Currently it is not precisely clear what are the pros and cons of ABAC over RBAC in general, and in smart home IoT in particular. To this end we provide an analysis of HABAC relative to the previously published EGRBAC (extended generalized role based access control) model for smart home IoT. We compare the theoretical expressive power of these models by providing algorithms for converting an HABAC specification to EGRBAC and vice versa, and discuss the insights for practical deployment of these models resulting from these constructions. We conclude that a hybrid model combining ABAC and RBAC features may be the most suitable for smart home IoT, and likely more generally.

ACM Reference Format:
Safwa Ameer and Ravi Sandhu. 2021. The HABAC Model for Smart Home IoT and Comparison to EGRBAC. In *Proceedings of the 2021 ACM Workshop on Secure and Trustworthy Cyber-physical Systems (SAT-CPS '21), April 28, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/SaT011f

1 Introduction and Motivation

The Internet of things (IoT) describes the network of physical objects (things) that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the Internet [1]. The concept of IoT has attracted many applications including consumer, organizational, industrial, infrastructure, and military applications. One of the most popular domains for deploying smart connected devices is the smart home. Surprisingly little attention has been paid to access control policy specification and authentication in home IoT. Home IoT is significantly different from traditional access control domains [21]. In home IoT, we have many users with complex social relationship who use the same devices. Moreover, the majority of smart home devices do not have screens and keyboards making them hands free for convenience while making access control more challenging. Real world examples of the shortcomings of current access control policy specification and authentication for home IoT devices have begun to appear [21, 22, 39]. The characteristics that make IoT distinct from prior computing domains necessitate a rethinking of access control and authentication [21].

In the literature, several access control models have been proposed for IoT in general. The majority of them are built on ABAC or RBAC. Some researchers argue that RBAC is more suitable for IoT since it is simpler in management and review, while ABAC is complex [3, 26, 27]. On the other hand, others argue that ABAC models are more scalable and dynamic, since they can capture different devices and environment contextual information [8, 9, 43]. However, RBAC models can be extended, such as the recent EGRBAC model [4] for smart home IoT which can express environment and device characteristics. RBAC enforcement may also be more lightweight for constrained home environment.

Hence, when it comes to smart homes, at this point it is not fully clear what is the benefit of ABAC over RBAC, and vice versa. Our intuitive insight is that a hybrid model will better capture smart home IoT AC requirements as this was already the case for traditional access control models. In order to further investigate this intuition our approach is to develop pure RBAC and pure ABAC based models explicitly defined to meet smart home challenges, and compare their benefits and drawbacks. This comparison will provide insights to guide us in designing "optimal" hybrid models in future work. Hence, in this paper, we propose smart home IoT ABAC (HABAC) model. Furthermore, we provide an analysis of HABAC relative to the previously published EGRBAC model [4]. We compare the theoretical expressive power of these models by providing algorithms to convert an HABAC specification to EGRBAC and vice versa. We discuss the insights for practical deployment of these models resulting from these constructions. The reasons for choosing EGRBAC in specific to compare it against HABAC are as follows. First EGRBAC is a contextual aware, fine grained model designed specifically to meet smart home challenges. Second, it satisfies the criteria for home IoT access control models

SAT-CPS '21, April 28, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8319-6/21/04…$15.00
https://doi.org/10.1145/3445969.3450428

proposed by [4]. Moreover, it meets the new perspective of smart home IoT access control requirements recently identified by [21].

The rest of the paper is organized as follows. Section 2 provides an analysis and review of related work, including an overview of EGRBAC [4]. Section 3 introduces the HABAC model along with a use case scenario. Section 4 shows how to translate EGRBAC policies into HABAC. An approach for constructing EGRBAC components and relations from a given HABAC specifications is described in Section 5. Section 6 discusses the insights of this work,including a comparison between HABAC and EGRBAC. Section 7 concludes the paper.

2 Related Work

Security researchers have carefully investigated smart IoT. While some have analyzed IoT security and privacy vulnerabilities [5, 12, 17, 41], others have studied IoT frameworks to identify security challenges, and design issues [14, 15, 20, 23]. It is generally accepted by most researchers that access control is a critical service in IoT. Ouaddah et al [33] provide an extensive survey on access control in IoT environments. Many access control solutions (user to device and/or device to device) have been proposed in the literature for different IoT applications. Some solutions are based on RBAC [16, 38] (as in [4, 6, 7, 9, 11, 26, 44]). While other solutions are based on ABAC [24, 25] (as in [8, 10, 19, 30, 42]. Some researchers argue that unlike ABAC, RBAC based model are incapable of capturing changing characteristics such as, environment attributes, device characteristics. However, the process of authorization is more simple in RBAC. As a result, some authors [27] proposed a combined access control model. Moreover, some of the proposed models in the literature are built on blockchain technology [2, 13, 31, 32]. However, as [31] described, the blockchain technology has some technical characteristics that could limit its applicability. For instance, cryptocurrency fees, and processing time. Few solutions have been proposed in the literature that are based on UCON [34, 35], for example [18, 28]. Several other access control models for IoT have been proposed, the authors in [3, 33, 36, 37] provided surveys on different access control models in the literature. However, none of them meet the criteria proposed by [4] for smart home AC models, and the new perspective for smart home IoT AC identified by [21]. Recently, Ameer et al [4] have proposed a criteria for home IoT access control models based on He et al analysis [21] and Ouddah et al survey [33]. Moreover, they introduced the EGRBAC model for smart home IoT access control, which is an RBAC based model that meets the characteristics proposed in both [4, 21]. Unlike traditional RBAC, EGRBAC captures the environment contextual changes, and different device characteristics. Hence, they confute the claim that RBAC based models are inadequate and inflexible in addressing the environment changes, and device or permission characteristics. That being so, it is not fully clear what is the benefit of ABAC over RBAC, and vice versa in the home IoT context. In this paper, we carefully investigated the expressiveness of HABAC compared to EGRBAC, and vice versa. We briefly review EGRBAC model below since it is relevant to Sections 4 and 5.

2.1 Background: EGRBAC For Smart Home IoT

Ameer et al introduced the extended generalized role based access control model [4]. In addition to the usual concept of User Roles, EGRBAC incorporates the notion of Device Roles and Environment Roles. It is illustrated in Figure 1. Roles (R) are analogous to the traditional RBAC users roles. However, in the context of smart homes, a role specifically represents the relationship between the user and the family. Device roles (DR) are defined as means of categorizing permissions of different devices. Environment roles (ER) represent environmental contexts, such as daytime/nighttime, and winter/summer. A role pair rp has a role part $rp.r$ that is the single role associated with rp, and an environment role part $rp.ER$ that is the subset of environment roles associated with rp. The main idea in EGRBAC as a whole is that a user is assigned a subset of roles and according to the current active roles in a session and the active environment roles, some role pairs will be active, whereby the user will get access to the permissions assigned to the device roles which are assigned to the current active role pairs.

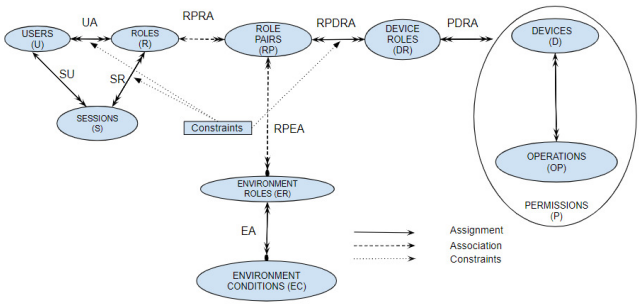

Figure 1: EGRBAC Model Components

3 HABAC Model for Smart Home IoT

ABAC models utilize attributes of users, sessions (subjects), objects, operations and environment to specify flexible, dynamic, and fine grained authorization policies. These characteristics arguably make ABAC suitable for deployment in complex domains such as smart home IoT. In this section, we define our HABAC (Home-IoT Attribute Based Access Control) model developed for user to device interaction in smart home IoT, where devices are the objects. The HABAC model is inspired by the ABAC model of Xin et al [25], extended to include environment attributes. Figure 2 depicts HABAC components. Users (U), Operations (OP), Devices (D), and Environment States (ES) are sets and shown in ovals. User Attributes (UA), Session Attributes (SA), Operation Attributes (OPA), Environment State Attributes (ESA), and Device Attributes (DA) are attribute functions and shown as squares. We have two types of constraints shown in rectangles: constraints on user attributes, and constraints on subject attributes. Table 1 formally defines these components. **Users (U)**: are humans interacting directly with the smart things. **Sessions (S)**: are similar to the concept of subjects in [25], users create sessions during which they may perform some actions in the system, the creating user is the only one who can terminate a session. **Devices (D)**: are smart home devices such as a smart light. **Operations (OP)**: represent actions on devices as specified by device manufacturers. **Environment States (ES)**: represent a picture of the environment at a given time instant that we want to describe, it may have values such as: current, yesterday, and so on. For simplicity, in this paper, the environment state is always equal to current. Users, devices, operations, and environment states

Table 1: HABAC Model Formalization

Basic Sets and Functions

$-U, S, OP, D$, and ES are finite sets of users, sessions, operations, devices, and environment states respectively.

$-UA, SA, OPA, ESA$, and DA are finite sets of users, sessions, operations, environment states, and devices attribute functions respectively.

– For each attribute att in $UA \cup SA \cup OPA \cup ESA \cup DA$, $Range(att)$ is the attribute range, a finite set of atomic values.

$-attType : UA \cup SA \cup OPA \cup ESA \cup DA \longrightarrow \{set, atomic\}$.

– Each attribute function ua, sa, opa, esa, and da in UA, SA, OPA, ESA, and DA respectively maps users in U, sessions in S, operation in OP, environment state in ES, and devices in D respectively to atomic or set attribute values. Formally:

$$ua : U \longrightarrow \begin{cases} Range(ua), & \text{if } attType(ua) = atomic \\ 2^{Range(ua)}, & \text{if } attType(ua) = set \end{cases}$$

$$sa : S \longrightarrow \begin{cases} Range(sa), & \text{if } attType(sa) = atomic \\ 2^{Range(sa)}, & \text{if } attType(sa) = set \end{cases}$$

$$opa : OP \longrightarrow \begin{cases} Range(opa), & \text{if } attType(opa) = atomic \\ 2^{Range(opa)}, & \text{if } attType(opa) = set \end{cases}$$

$$esa : ES \longrightarrow \begin{cases} Range(esa), & \text{if } attType(esa) = atomic \\ 2^{Range(esa)}, & \text{if } attType(esa) = set \end{cases}$$

$$da : D \longrightarrow \begin{cases} Range(da), & \text{if } attType(da) = atomic \\ 2^{Range(da)}, & \text{if } attType(da) = set \end{cases}$$

$-sessionUser : S \longrightarrow U$, maps each session to its creator user.

$-sa(s) \subseteq ua(sessionUser(s))$, for each $s \in S$

User Attributes Constraints

– User attribute constraint $UAConstraint \subseteq UAP \times 2^{UAP}$, where user attribute pair UAP is a pair of user attribute, and its value (ua_{name}, ua_{value}).

– Each $uac = ((ua_x, y), UAP_j) \in UAConstraint$ specifies the following invariant:

$$\begin{cases} (\forall u_l \in U)(\forall (ua_m, n) \in UAP_j)[ua_x(u_l) = y \Rightarrow ua_m(u_l) \neq n], & \text{if } attType(ua_x) = attType(ua_m) = atomic \\ (\forall u_l \in U)(\forall (ua_m, n) \in UAP_j)[y \in ua_x(u_l) \Rightarrow n \notin ua_m(u_l)], & \text{if } attType(ua_x) = attType(ua_m) = set \end{cases}$$

Subject Attributes Constraints

– Subject attribute constraint $SAConstraint \subseteq SAP \times 2^{SAP}$, where subject attribute pair SAP is a pair of subject attribute, and its value (sa_{name}, sa_{value}).

– Each $sac = ((sa_x, y), SAP_j) \in SAConstraint$ specifies the following invariant:

$$\begin{cases} (\forall s_l \in S)(\forall (sa_m, n) \in SAP_j)[sa_x(s_l) = y \Rightarrow sa_m(s_l) \neq n], & \text{if } attType(sa_x) = attType(sa_m) = atomic \\ (\forall s_l \in S)(\forall (sa_m, n) \in SAP_j)[y \in sa_x(s_l) \Rightarrow n \notin sa_m(s_l)], & \text{if } attType(sa_x) = attType(sa_m) = set \end{cases}$$

Authorization Functions (Policies)

– Operation authorization function: For each $op \in OP$, $Authorization_{op}(s : S, es : ES, d : D)$ is a propositional logic formula returning true or false, and it is defined using the following policy language:

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

Authorization Decision

– A subject $s \in S$ is allowed to perform an operation $op \in OP$ in a device $d \in D$ during an environment state $es \in ES$, if the attributes of subject s, operation op, device d, and environment state es satisfy $Authorization_{op}(s : S, es : ES, d : D)$ policies.

have characteristics which are used in access control decision and expressed as their attributes. An attribute is a function which takes an entity such as a user and returns a specific value from its range. An attribute range is given by a finite set of atomic values. An atomic valued attribute will return one value from the range, while a set valued attribute will return a subset of the range. In general, we have two types of attributes: (a) Dynamic Attributes, this type is rapidly changing due to different conditions. For instance, user location, weather, and device temperature. (b) Static Attributes, this type is relatively static and need administrator action to be changed. For example, user relationship, device level of danger, and so on. **User Attributes (UA)**: is the set of attributes associated with users. **Session Attributes (SA)**: is the set of attributes associated with sessions. Session attributes are set by the creating user, and are constrained by policies established by security architects (discussed later). Session attributes require an initial value at creation time which will be inherited from the corresponding creator user's attributes. **Operation Attributes (OPA)**: is the set of attributes assigned to different operations, for example you may want to characterize dangerous operations by creating an operation attribute called "Dangerous Operations" and associate it with those operations. **Device Attributes (DA)**: is the set of attributes associated with smart things, for instance, "Kitchen devices", and "Alex devices". **Environment state Attributes (ESA)**: is the set of attributes describe the environment condition of a specific time. For example, "days of the week", and "weather condition". How different environment state attributes get triggered is outside the scope of our model. Operation, and device attributes are partial function s; we may have some devices, or some operations that are not assigned to some attributes. On the other hand, users, sessions, and environment roles attributes are total functions. A constraint is an invariant that must be maintained at all times. In HABAC

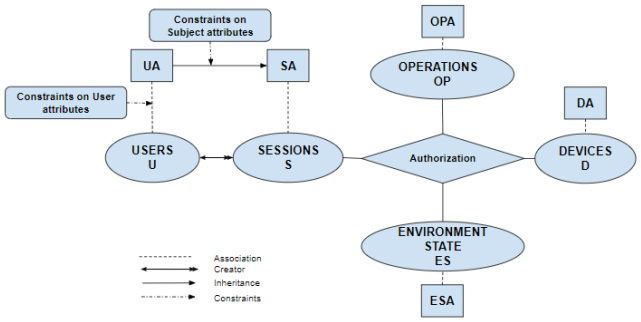

Figure 2: Smart Home IoT ABAC Model

we define two types of constraints, as follows. **Constraints on User attributes**: these constraints enforce restrictions on user attributes. For instance, if we have the following user attribute constraint $((Relationship, kid), \{(Adults, True)\})$, this constraint implies that if a user *Relationship* attribute is equal to *kid*, then he cannot have $True$ as the value of the attribute function *Adults*. **Constraints on Subject attributes**: these constraints enforce restrictions on subject attributes. For instance, if we have the following subject attribute constraint: $((Relationship, staying\ Home\ Kid), \{(Relationship, travel\ Abroad\ Kid)\})$, this constraint implies that for any session s_l if $staying\ Home\ Kid \in Relationship(s_l)$, then $travel\ Abroad\ Kid \notin Relationship(s_l)$. **Authorization Functions (Policies)**: are two-valued boolean function which are evaluated for each access decision. An authorization policy for a specific operation takes a subject, an environment state, a device and returns true or false based on attribute values. In Table 1 we define a common policy language for authorization functions using propositional logic formula.

3.1 HABAC Use Case

We present a use case to demonstrate the components and configurations of HABAC. The objective is as follow: (a) Allow kids to use kids friendly operations in entertainment devices (*G*, *PG* contents in TV, and PS_{A3} (games for group age below 3 years old), PS_{A7} (games for group age below 7 years old) contents in Play Station) during specific time (weekends afternoons and evenings, and weekdays evenings). (b) Allow teenagers to use entertainment devices unconditionally. (c) Authorize teenagers to use dangerous kitchen devices (Oven) only when one of the parents is in the kitchen. (d) Authorize teenagers to use non dangerous kitchen devices (Fridge) unconditionally. (e) Allow parents to use any operation in any device unconditionally. To achieve these objectives, HABAC can be configured as shown in Table 2. Here, we have three users, *alex*, *bob*, and *anne* with user attribute *Relationship* set to the values *kid*, *teenager*, and *parent* respectively. When a user u_x creates a session, it will automatically inherit subset of u_x attributes. We have five devices *TV*, *PlayStation*, *Oven*, *Fridge* and *FrontDoor*. *Oven*, and *Fridge* are assigned the following device attributes by the house owner $Fridge \leftarrow (DangerouseKitchenDevices : False)$, and $Oven \leftarrow (DangerouseKitchenDevices : True)$. We have twelve operations *G*, *PG*, *A*3, *A*7, *A*12, *BuyGames*, *ON*, *OFF*, *Open*, *Close*, *Lock*, and *Unlock*. These operations are assigned to operation attributes as following, (*G*, *A*3, and *A*7) $\leftarrow (KidsFriendly : True)$, while (*PG*, *A*12, *BuyGames*) $\leftarrow (KidsFriendly : False)$. We have one environment state *current* which has three attribute functions (*day*, *time*, and *ParentInKitchen*). The authorization function is a disjunction of five propositional statements. The first statement gives kids access to kids friendly operations during weekdays evenings, or weekends afternoon and evenings. The second statement gives teenagers access to dangerous kitchen devices only when one of the parents is in the kitchen. The third statement authorizes teenagers to use non dangerous kitchen devices unconditionally. The fourth statement allows teenagers to access kids friendly operations, and non kids friendly operations unconditionally. Finally, the fifth statement gives parents access to anything unconditionally.

Table 2: HABAC Use Case

$U = \{alex, bob, anne\}$,
$UA = \{Relationship, Location\}$
$Relationship : u : U \rightarrow \{parent, kid, teenager\}$
$Location : u : U \rightarrow \{Kitchen, MasterBedRoom, BedRoom_1, BedRoom_2, LivingRoom\}$
$Relationship(alex) = kid$
$Relationship(anne) = teenager$
$Relationship(bob) = parent$

$S = \{\ldots\}$
$SA = \{Relationship, Location\}$
$Relationship : s : S \rightarrow \{parent, kid, teenager\}$
$Location : s : S \rightarrow \{Kitchen, MasterBedRoom, BedRoom_1, BedRoom_2, LivingRoom\}$

$D = \{TV, PlayStation, Oven, Fridge, FrontDoor\}$
$DA = \{DangerouseKitchenDevices\}$
$DangerouseKitchenDevices : d : D \rightarrow \{True, False\}$
$DangerouseKitchenDevices(Oven) = True$
$DangerouseKitchenDevices(Fridge) = False$

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

$OP_{TV} = \{G, PG, \ldots\}$
$OP_{PlayStation} = \{A3, A7, A12, BuyGames, \ldots\}$
$OP_{Oven} = \{ON, OFF\}$
$OP_{Fridge} = \{Open, Close\}$
$OP_{FrontDoor} = \{Lock, Unlock\}$
$OP = OP_{TV} \cup OP_{PlayStation} \cup OP_{Oven} \cup OP_{Fridge} \cup OP_{FrontDoor}$
$OPA = \{KidsFriendly\}$
$KidsFriendly : op : OP \rightarrow \{True, False\}$
$KidsFriendly(G) = KidsFriendly(A3) = KidsFriendly(A7) = True$
$KidsFriendly(PG) = KidsFriendly(A12) = KidsFriendly(BuyGames) = False$

$Authorization_{op}(s : S, es : ES, d : D) \equiv$
$(Relationship(s) = kid \wedge ((day(current) \in \{Sa, S\} \wedge 12:00 \leq time(current) \leq 19:00) \vee (day(current) \in \{M, T, W, Th, F\} \wedge 17:00 \leq time(current) \leq 19:00)) \wedge KidsFriendly(op) = True \vee$
$(Relationship(s) = teenager \wedge ParentInKitchen(current) = True \wedge DangerouseKitchenDevices(d) = True) \vee$
$(Relationship(s) = teenager \wedge DangerouseKitchenDevices(d) = False) \vee$
$(Relationship(s) = teenager \wedge (KidsFriendly(op) = True \vee KidsFriendly(op) = False)) \vee$
$(Relationship(s) = parent)$

Table 3: EGRBAC Configuration in HABAC

- $U_{HABAC} = U_{EGRBAC}$
- $UA = SA = \{Relationship\}$
- $Relationship : u \in U_{HABAC} \rightarrow 2^R$
- $Relationship : s \in S \rightarrow 2^R$
- $(\forall u_i \in U_{HABAC})[Relationship(u_i) = \{r_x | (u_i, r_x) \in UA\}]$

- $UAConstraint = \{uac_i\}$
- For all $ssdc_i = (r_i, R_j) \in SSDConstraints$:
$uac_i = ((Relationship, r_i), UAP_j)$, where $UAP_j = \{(Relationship, r_n) | r_n \in R_j\}$

- $SAConstraint = \{sac_i\}$
- For all $dsdc_i = (r_i, R_j) \in DSDConstraints$:
$sac_i = ((Relationship, r_i), SAP_j)$, where $SAP_j = \{(Relationship, r_n) | r_n \in R_j\}$

- $ES = \{Current\}$
- $ESA = ER$
$(\forall esa_i \in ESA)[esa_i : es \in ES \rightarrow \{True, False\}]$

- $D_{HABAC} = D_{EGRBAC}$, $OP_{HABAC} = OP_{EGRBAC}$
- $DA = OPA = DR$
- $(\forall da_i \in DA)\ [da : d \in D_{HABAC} \rightarrow \{True, False\}]$
- $(\forall opa_i \in OPA)[opa : op \in OP_{HABAC} \rightarrow \{True, False\}]$
- $(\forall (dr_y \in DR, p_x \in \{p_i | (p_i, dr_y) \in PDRA\}))[dr_y(p_x.op) = True, dr_y(p_x.d) = True]$
- $(\forall (dr_y \in DR, p_y \in \{p_j | (p_j, dr_y) \notin PDRA\}))[dr_y(p_x.op) = False, dr_y(p_x.d) = False]$

- For each $rpdra_i = ((r_i, ER_i), dr_i) \in RPDRA$, we construct an authorization policy as following:
$[Authorization_{op}(s : S, es : ES, d : D) \equiv Relationship(s) = r_i \wedge dr_i(op) = True \wedge (\bigcap_{esa \in ER_i} esa(current) = True) \wedge dr_i(d) = True]$
- The final authorization policy is the disjunction of every created authorization policy.

4 Constructing HABAC From EGRBAC

In this section, we take a further step toward comparing HABAC, and EGRBAC. We introduce HABAC configuration that translates EGRBAC policies in a manner that they can be implemented by HABAC. The purpose is to see whether we can fully express any EGRBAC configuration in HABAC model, and if not which model is more expressive, and in what terms. The configuration of HABAC for a given EGRBAC configuration is shown in Table 3. The goal is to construct HABAC elements ($U, UA, SA, ES, ESA, D, DA, OP, OPA$) and the authorization policy function from EGRBAC policy in such a way that the authorizations are the same as those under EGRBAC. The inputs are EGRBAC component sets $R, U_{EGRBAC}, UA, EC, ER, EA, DR, PDRA, P_{EGRBAC}, D_{EGRBAC}, OP_{EGRBAC}$. The outputs are $U_{HABAC}, UA, ES, ESA, OP_{HABAC}, OPA, D_{HABAC}, DA$, and the authorization policy function $Authorization_{op}(s : S, es : ES, d : D)$. The set of users, devices, and operations are the same in both systems. Roles are expressed through the user attribute *Relationship* in HABAC. *Relationship* is a user and a subject attribute that takes a user or a subject as an input and returns the set of roles assigned to that user or that subject. Static separation of duty constraints *SSDConstraints* are translated into user attributes constraints in HABAC. Dynamic separation of duty constraints *DSDConstraints* are translated into subject attributes constraints in HABAC. Environment roles are translated into atomic environment state attributes. How to trigger different environment states attributes in response to environment's changes is outside the scope of this model. In EGRBAC device roles are ways of categorizing permissions. Since we do not have permission attributes in HABAC, and since a permission is mapping between a device and an operation, we translate device roles in EGRBAC into atomic operation attributes and atomic device attributes with a range of values equal to $\{True, False\}$. The final step is to translate the authorization policies. In EGRBAC it is the *RPDRA* that gives specific role pairs and hence users access to specific device roles and hence permissions. Therefore, we translate each $rpdra_i = ((r_i, ER_i), dr_i) \in RPDRA$ into an authorization policy. The final authorization policy is the disjunction of every created authorization policy. As a result, we have only one user attribute which is *Relationship*. The number of user attributes constraints is equal to the number of *SSDConstraints*. The number of subject attributes constraints is equal to the number of *DSDConstraints*. The number of operation attributes, and the number of device attributes are equal to the number of device roles. The number of environment state attributes is equal to the number of environment roles. In HABAC we can not create something equivalent to EGRBAC *PRConstraints*. As shown in Section 2.1, in EGRBAC, the way a user get access to specific set of permissions happens through a series of assignments, the most critical assignment is the *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 on the other hand, we don't have similar "attack point" that can be controlled to prevent specific access permission. The only way to make sure that certain users can not get access to specific permissions is by checking each request to access these permissions and look for those prohibited users. This is a significant advantage of EGRBAC in which we can enforce such constraints at assignment time whereas HABAC-like models would need to enforce these at enforcement time. To summarize, the construction of HABAC shown in Table 3 is equivalent to the given EGRBAC configuration including static and dynamic separation of duty constraints. The claim of equivalence is intuitively obvious since the construction is effectively one for one and straightforward. A formal argument can be presented along the lines of [40] but is tedious and does not provide meaningful insight.

5 Constructing EGRBAC from HABAC

In this section, we introduce our methodology to construct EGRBAC components and configurations from HABAC policy configuration. Our EGRBAC constructing approach works perfectly for HABAC policies that contain environment attributes, and static sessions, operations, and devices attributes. It can also handles policies that do not involve comparing two different types of attributes. Furthermore, we found that due to some limitations in EGRBAC, it is either not possible to capture policies involving dynamic sessions, operations, and devices attributes, or costly (leads to role explosion). We followed a bottom-up role engineering approach. Traditional algorithms for translating ABAC systems into RBAC using bottom-up approach, first represents the ABAC system in UPA matrix, and then do some sort of role mining. The rows and columns of the

matrix correspond to users and permissions, respectively. If a user is assigned a particular permission, the corresponding entry of the matrix contains a 1; otherwise, it contains a 0. From this perspective, role engineering is a process of matrix decomposition, wherein the Boolean matrix UPA is decomposed into two Boolean matrices (UA and PA), which together give the original access control policy [29]. EGRBAC is a more sophisticated model than RBAC; since in addition to user roles, it captures environment, and device characteristics through environment roles, and device roles respectively. In EGRBAC we do not give access to permissions directly, instead we give access to device roles. Hence, instead of UPA matrix, we first need to construct a user-device role assignment matrix, we call it user device role assignment array $UDRAA$. Moreover, users do not get access to authorized device roles unconditionally, some times specific set of environment roles need to be active. $UDRAA$ need to capture these information; each cell (u_i, dr_j) in $UDRAA$ contains zero if u_i cannot access dr_j, contains one if u_i can access dr_j unconditionally, or contains C if u_i can access dr_j when the set of conditions C is satisfied, were C is a set of session conditions (for example role), and environment conditions.

5.1 From Authorization policy to Authorization Array

An authorization policy is expressed in a logical clause. First we convert it into a disjunctive normal form (DNF). All logical formulas can be converted into an equivalent DNF form. Here is the authorization policy of our HABAC use case which introduced in Table 2 after following the standard approach to convert it into a DNF format.

$Authorization_{op}(s : S, es : ES, d : D) \equiv$
$(Relationship(s) = kid \wedge\ day(current) \in \{Sa, S\} \wedge 12:00 \leq time(current) \leq 19:00 \wedge KidsFriendly(op) = True) \vee$
$(Relationship(s) = kid \wedge\ day(current) \in \{M, T, W, Th, F\} \wedge 17:00 \leq time(current) \leq 19:00 \wedge KidsFriendly(op) = True) \vee$
$(Relationship(s) = teenager \wedge ParentInKitchen(current) = True \wedge DangerouseKitchenDevices(d) = True) \vee$
$(Relationship(s) = teenager \wedge DangerouseKitchenDevices(d) = False) \vee$
$(Relationship(s) = teenager \wedge KidsFriendly(op) = False) \vee$
$(Relationship(s) = teenager \wedge KidsFriendly(op) = True) \vee$
$(Relationship(s) = parent)$.

We call each conjuncted term a condition. We have session, environment, device, operation, and mix conditions which are conditions that involve session, environment, device, operation, and any two types of attributes respectively. In the DNF authorization policy of our HABAC use case mentioned above we have seven conjunctive clauses, each conjunctive clause represents one access authorization rule. To construct the authorization array we evaluate every $u_i \in U$, $d_j \in D$, and $op_k \in OP$ combination against each conjunctive clause, whenever a combination satisfies every term (condition) in a conjunctive clause except those conditions which involve environment state attributes, we create a raw $(u_i, d_j, op_k, current, C)$ for that combination in the authorization array. Where, C is the set of session and environment related conditions in the examined conjunctive clause. **Authorizations array (AA):** an authorization raw $(u_i, d_j, op_k, es_l, C)$ denotes that the user u_i is allowed to perform an operation op_k on a device d_j during the environment state es_l whenever the set of environment and session conditions in C are satisfied. The AA of the use case in Table 2 is shown in Table 4. For illustration purposes each different color presents the authorization fields for different user.

Table 4: AA for HABAC Use Case

User u	Device d	Operation op	Environment state es	Conditions C
alex	TV	*G*	current	X
alex	PS	*A3*	current	X
alex	PS	*A7*	current	X
alex	TV	*G*	current	Z
alex	PS	*A3*	current	Z
alex	PS	*A7*	current	Z
bob	TV	*G*	current	{*Relationship*(*s*) = *parent*}
bob	TV	*PG*	current	{*Relationship*(*s*) = *parent*}
bob	PS	*A3*	current	{*Relationship*(*s*) = *parent*}
bob	PS	*A7*	current	{*Relationship*(*s*) = *parent*}
bob	PS	*A12*	current	{*Relationship*(*s*) = *parent*}
bob	PS	*BuyGames*	current	{*Relationship*(*s*) = *parent*}
bob	Oven	*ON*	current	{*Relationship*(*s*) = *parent*}
bob	Oven	*OFF*	current	{*Relationship*(*s*) = *parent*}
bob	Fridge	*Open*	current	{*Relationship*(*s*) = *parent*}
bob	Fridge	*Close*	current	{*Relationship*(*s*) = *parent*}
bob	FrontDoor	*Lock*	current	{*Relationship*(*s*) = *parent*}
bob	FrontDoor	*Unlock*	current	{*Relationship*(*s*) = *parent*}
anne	TV	*G*	current	{*Relationship*(*s*) = *teenager*}
anne	TV	*PG*	current	{*Relationship*(*s*) = *teenager*}
anne	PS	*A3*	current	{*Relationship*(*s*) = *teenager*}
anne	PS	*A7*	current	{*Relationship*(*s*) = *teenager*}
anne	PS	*A12*	current	{*Relationship*(*s*) = *teenager*}
anne	PS	*BuyGames*	current	{*Relationship*(*s*) = *teenager*}
anne	Oven	*ON*	current	Y
anne	Oven	*OFF*	current	Y
anne	Fridge	*OPEN*	current	{*Relationship*(*s*) = *teenager*}
anne	Fridge	*CLOSE*	current	{*Relationship*(*s*) = *teenager*}

X = $\{Relationship(s) = kid\ ,\ day(current) \in \{Sa, S\}, 12:00 \leq time(current) \leq 19:00\}$.
Y = $\{Relationship(s) = teenager\ ,\ ParentInKitchen(current) = True\}$.
Z= $\{Relationship(s) = kid\ ,\ day(current) \in \{M, T, W, Th, F\}, 17:00 \leq time(current) \leq 19:00\}$.

Table 5: PDRA array for HABAC Use Case

	DangerouseKitchenDevices = True	DangerouseKitchenDevices = False	KidsFriendly = True	KidsFriendly = False	RemPerm
(*TV*, *G*)	0	0	1	0	0
(*TV*, *PG*)	0	0	0	1	0
(*PlayStation*, *A3*)	0	0	1	0	0
(*PlayStation*, *A7*)	0	0	1	0	0
(*PlayStation*, *A12*)	0	0	0	1	0
(*PlayStation*, *BuyGames*)	0	0	0	1	0
(*Oven*, *ON*)	1	0	0	0	0
(*Oven*, *OFF*)	1	0	0	0	0
(*Fridge*, *Open*)	0	1	0	0	0
(*Fridge*, *Close*)	0	1	0	0	0
(*FrontDoor*, *Lock*)	0	0	0	0	1
(*FrontDoor*, *Unlock*)	0	0	0	0	1

Table 6: UDRAA for HABAC Use Case

	DangerouseKitchenDevices = True	DangerouseKitchenDevices = False	KidsFriendly = True	KidsFriendly = False	RemPerm
alex	0	0	{*X*, *Z*}	0	0
bob	{{*Relationship*(*s*) = *parent*}}	{{*Relationship*(*s*) = *parent*}}	{{*Relationship*(*s*) = *parent*}}	{{*Relationship*(*s*) = *parent*}}	{{*Relationship*(*s*) = *parent*}}
anne	{*Y*}	{{*Relationship*(*s*) = *teenager*}}	{{*Relationship*(*s*) = *teenager*}}	{{*Relationship*(*s*) = *teenager*}}	0

X = $\{Relationship(s) = kid\ ,\ day(current) \in \{Sa, S\}, 12:00 \leq time(current) \leq 19:00\}$.
Y = $\{Relationship(s) = teenager\ ,\ ParentInKitchen(current) = True\}$.
Z= $\{Relationship(s) = kid\ ,\ day(current) \in \{M, T, W, Th, F\}, 17:00 \leq time(current) \leq 19:00\}$.

5.2 Approach

The goal is to construct EGRBAC elements (U, R, EC, ER, RP, D, OP, P, DR), assignments (UA, EA, $PDRA$, $RPDRA$), and associations ($RPRA$, $RPEA$) from HABAC policies in such a way that the authorizations are the same as those under HABAC. The inputs are HABAC set of users U_{HABAC}, set of devices D_{HABAC}, set of operations OP_{HABAC}, UA, SA, ESA, OPA, DA, and the authorization

array AA. The outputs are EGRBAC components U, R, UA, EC, ER, EA, RP, $RPRA$, $RPEA$, D, OP, P, DR, $PDRA$, and $RPDRA$. The steps are following:

Step 1: Initialization. The set of users, devices, and operations are the same in both systems, hence $U = U_{HABAC}$, $D = D_{HABAC}$, and $OP = OP_{HABAC}$. For every operation op_i, and device d_j pair, where op_i is assigned to d_j by the device manufacturers, create a permission.

Step 2: Create the set of device roles DR. (a) Create a device role dr for each operation attribute instance, or device attribute instance. DR here are represented as a condition of the form $opa = x$, or $da = x$. Where x is an instance of the attribute value. (b) Create one device role call it remaining permissions $RemPerm$ for all the permissions $p_l = (d_i, op_j)$, where d_i is not assigned to any device attributes, and op_j is not assigned to any operation attribute. This device role captures the cases where some users have access to specific permissions directly without involving device's or operation's attributes.

Step 3: Construct the permission device role assignment array $PDRA$. It is a many-to-many mapping of P set and DR set (constructed in Step 2). To construct $PDRA$ we first make a column for each DR, and make a row for each permission p_x. Then, we fill the array $PDRA$, where $PDRA[i, j] = 1$ in two cases, first if for the permission p_i (corresponding to the row i) $p_i.op$ or $p_i.d$ satisfies the condition corresponding to the device role of the column j dr_j. Second, if $p_i.op$ is not assigned to any operation attribute, and $p_i.d$ is not assigned to any device attribute, and the device role corresponding to this column is $RemPerm$. $PDRA[i, j] = 0$ otherwise. For every $PDRA[i, j] = 1$, add the pair (p_i, dr_j) to the set $PDRA$ of EGRBAC. See Table 5 for the $PDRA$ array of our HABAC use case.

Step 4: Construct the user device role authorization array $UDRAA$ from the authorization array AA, and $PDRA$. $UDRAA \subseteq U \times DR$, a many to many mapping between U and DR. To construct $UDRAA$, we first make a raw for each user, and a column for each device role. Then, for every $UDRAA[i, j] \in UDRAA$ we check the AA for every u_i, and p_x combination, where $(p_x, dr_j) \in PDRA$. Here, we have three cases: (a) $UDRAA[i, j] = 1$ if user u_i can access all the permissions assigned to the device role dr_j without any condition. (b) $UDRAA[i, j] = 2^C$, where C is a set of session, and environment conditions that need to be satisfied together for a u_i to access all the permissions assigned to dr_j. Note that these sets of conditions have to be the same for each permission assigned to dr_j. (c) Finally, $UDRAA[i, j] = 0$ if user u_i is not allowed to access all the permissions assigned to the device role dr_j, or is allowed to access different permissions in dr_j but under different set of conditions. Table 6 shows $UDRAA$ for our use case.

Step 5: Construct the rest of EGRBAC elements (R, EC, ER, RP), assignments $(UA, EA, RPDRA)$, and associations $(RPRA, RPEA)$ by following our proposed EGRBAC users and environment roles constructing algorithm introduced in Section 5.3. The set of users roles R constructed here is the set of candidate users roles.

Step 6: Merge similar users roles. To do so, we run our developed role merging algorithm illustrated in Section 5.4.

Algorithm 1 EGRBAC Users and Environment Roles Construction

Require: $UDRAA$
Require: $ColumnDR(j)$: return the device role corresponding to the column j in $UDRAA$.
Require: $RawUser(i)$: return the user corresponding to the raw i in $UDRAA$.
Require: $ContainsSC(X)$: Return True if the set of conditions X contains at least one session condition.
Require: $ContainsESC(X)$: Return True if the set of conditions X contains at least one environment state condition.
Require: $IsESC(c)$: Return True if c is an environment state condition.
Require: $ToString(x)$: Return the value of x in a string format.

Initilize n = Number of users, m = Number of device roles, $R = \{\}$, $UA = \{\}$, $EC = \{\}$, $ER = \{\}$, $EA = \{\}$, $RP = \{\}$, and $RPDRA = \{\}$,
for $j \leftarrow 1$ to m **do**
  for $i \leftarrow 1$ to n **do**
    if $UDRAA[i, j] = 1$ **then**
      $er_x = Any\_Time$, $ec_x = True$
      $EC = EC \cup \{ec_x\}$, $ER = ER \cup \{er_x\}$
      $EA = EA \cup \{(\{ec_x\}, er_x)\}$
      $SER = \{er_x\}$
      $r_m = ToString(ColumnDR(j))$
      $R = R \cup \{r_m\}$
      $RP = RP \cup \{rp_z\}$, where $rp_z.r = r_m, rp_z.ER = SER$
      $RPDRA = RPDRA \cup \{(rp_z, ColumnDR(j))\}$
      $UA = UA \cup \{(RawUser(i), r_m)\}$
    else if $UDRAA[i, j] \neq 1 \wedge UDRAA[i, j] \neq 0$ **then**
      for each $X \in UDRAA[i, j]$ **do**
        if $(\neg ContainsSC(X) \wedge ContainsESC(X)$ **then**
          $SER = \{\}$
          for each $y \in X$ **do**
            Create ec_y, and er_y
            $EC = EC \cup \{ec_y\}$, $ER = ER \cup \{er_y\}$
            $EA = EA \cup \{(\{ec_y\}, er_y)\}$
            $SER = SER \cup \{er_y\}$
          end for
          $r_m = ToString(ColumnDR(j))$ +" $\wedge$ "+ $ToString(X)$
          $R = R \cup \{r_m\}$
          $RP = RP \cup \{rp_z\}$, where $rp_z.r = r_m, rp_z.ER = SER$
          $RPDRA = RPDRA \cup (rp_z, ColumnDR(j))$
          $UA = UA \cup \{(RawUser(i), r_m)\}$
        else if $(ContainsSC(X) \wedge \neg ContainsESC(X)$ **then**
          $er_x = Any\_Time$, $ec_x = True$
          $EC = EC \cup \{ec_x\}$, $ER = ER \cup \{er_x\}$
          $EA = EA \cup \{(\{ec_x\}, er_x)\}$
          $SER = \{er_x\}$
          $r_m = ToString(ColumnDR(j))$ +" $\wedge$ "+ $ToString(X)$
          $R = R \cup \{r_m\}$
          $RP = RP \cup \{rp_z\}$, where $rp_z.r = r_m, rp_z.ER = SER$
          $RPDRA = RPDRA \cup (rp_z, ColumnDR(j))$
          $UA = UA \cup \{(RawUser(i), r_m)\}$
        else if $(ContainsSC(X) \wedge ContainsESC(X))$ **then**
          $SER = \{\}$
          for each $y \in \{y | (y \in X) \wedge IsESC(y)\}$ **do**
            Create ec_y, and er_y
            $EC = EC \cup \{ec_y\}$, $ER = ER \cup \{er_y\}$
            $EA = EA \cup \{(\{ec_y\}, er_y)\}$
            $SER = SER \cup \{er_y\}$
          end for

r_m = $ToString(ColumnDR(j))$ + " ∧ " + $ToString(X)$
$R = R \cup \{r_m\}$
$RP = RP \cup \{rp_z\}$, where $rp_z.r = r_m, rp_z.ER = SER$
$RPDRA = RPDRA \cup (rp_z, ColumnDR(j))$
$UA = UA \cup \{(RawUser(i), r_m)\}$
end if
end for
end if
end for
end for

5.3 EGRBAC Users and Environment Roles Constructing Algorithm:

The goal is to Construct EGRBAC elements (R, EC, ER, RP), assignments ($UA, EA, RPDRA$), and associations ($RPRA, RPEA$) from $UDRAA$. See Algorithm 1 for the full algorithm. The input is $UDRAA$. The outputs are R, UA, EC, ER, EA, RP, and $RPDRA$. The steps are shown as following:

Step 1: Initialize the following EGRBAC sets $R = \{\}, UA = \{\}, EC = \{\}, ER = \{\}, EA = \{\}, RP = \{\}, RPDRA = \{\}$, and the following constants m = Number of device roles, n = Number of users.

Step 2: Loop through the columns of $UDRAA$, Table 6 for our HABAC use case. Each column is corresponding to users access rights to a specific device role. Inside each column loop through the fields of different rows. Here we have two cases:

A. $UDRAA[i, j] = 1$**,** according to the way $UDRAA$ was constructed this means the user corresponding to this raw u_i can access the device role of this column dr_j unconditionally. In this case, the algorithm does the following:

a Creates an environment role $er_x = Any\_Time$ and add it to the set ER, an environment condition $ec_x = True$ and add it to the set EC. Add $(\{ec_x\}, er_x)$ to the set EA, this implies that the environment role Any_Time will always be active. Create a set of environment roles SER and add er_x to it $SER = \{er_x\}$.
b Creates a role $r_m = ToString(ColumnDR(j))$ which corresponds to accessing this column device role anytime, and unconditionally. Add this role to the set R.
c Defines a role pair rp_z, where $rp_z.r = r_m$ and $rp_z.ER = SER$. Add rp_z to the set RP.
d Assigns the role pair rp_z to the device role corresponding to this column by adding the pair $(rp_z, ColumnDR(j))$ to the set $RPDRA$.
e Assigns the role r_m to the user corresponding to this raw by adding the pair$(RawUser(i), r_m)$ to the set UA.

B. $UDRAA[i, j] \neq 1 \wedge UDRAA[i, j] \neq 0$**,** which means that the user u_i can access the device role dr_j under specific set of user and environment conditions defined by $UDRAA[i, j]$. Here, $UDRAA[i, j]$ is a set of sets of conditions, each set of conditions define a group of session, and environment conditions that need to be satisfied together in order for the user u_i to be able to access the device role dr_j. Loop through each set of conditions $X \in UDRAA[i, j]$, X satisfies only one of the three following options:

(1) X contains environment conditions only. In this case the algorithm first loops through each environment condition to create a corresponding environment condition ec_y, environment role er_y, and add these environment roles to the set SER. Second, the algorithm creates a corresponding role which represents accessing this column device role when the set of environment attributes conditions that form X is satisfied. Finally, it follows the same three steps c, d, and e explained in the previous case (when $UDRAA[i, j] = 1$).
(2) X contains session conditions only. Here the algorithms follows the same steps explained in case A (when $UDRAA[i, j] = 1$). The only difference here is that the created role corresponds to access this column device role anytime, under the set of user conditions expressed by X instead of unconditionally as in case A.
(3) X contains session and environment conditions. In this case the algorithm creates corresponding environmnet roles, environment conditions, and user role. It then follows the same three step c, d, and e explained in case A.

5.4 Users Roles Merging Algorithm

The main purpose of this algorithm is to merge roles that have similar users assignments. For each two roles r_i, r_j which are assigned to the same set of users, the algorithm does the following: (a) For every role pair rp_k, in which the role part of it $rp_k.r$ is equal to r_i, change the role part of it to r_j ($rp_k.r = r_j$). (b) Remove r_i from the set of roles R. (c)For every $(u_l, r_i) \in UA$, remove the pair (u_l, r_i) from the set UA. See Algorithm 2 for the complete algorithm. After applying the first five steps of the approach of constructing EGRBAC from HABAC introduced in Section 5.2 on our HABAC use case, we will end up having eleven roles as following: $R = \{$
$r_1 \equiv DangerouseKitchenDevices = True \wedge Relationship(s) = parent$,
$r_2 \equiv DangerouseKitchenDevices = True \wedge \{Relationship(s) = teenager, ParentInKitchen(current) = True\}$,
$r_3 \equiv DangerouseKitchenDevices = False \wedge Relationship(s) = parent$,
$r_4 \equiv DangerouseKitchenDevices = False \wedge Relationship(s) = teenager$,
$r_5 \equiv KidsFriendly = True \wedge \{Relationship(s) = kid,$
$day(current) \in \{Sa, S\}, 12:00 \leq time(current) \leq 19:00\}$,
$r_6 \equiv KidsFriendly = True \wedge \{Relationship(s) = kid,$
$day(current) \in \{M, T, W, Th, F\}, 17:00 \leq time(current) \leq 19:00\}$,
$r_7 \equiv KidsFriendly = True \wedge Relationship(s) = parent$,
$r_8 \equiv KidsFriendly = True \wedge Relationship(s) = teenager$,
$r_9 \equiv KidsFriendly = False \wedge Relationship(s) = parent$,
$r_{10} \equiv KidsFriendly = False \wedge Relationship(s) = teenager$,
$r_{11} \equiv RemPerm \wedge Relationship(s) = parent\}$.
These roles will be assigned to different users as following: $UA = \{(alex, r_5), (alex, r_6), (bob, r_1), (bob, r_3), (bob, r_7), (bob, r_9), (bob, r_{11}), (anne, r_2), (anne, r_4), (anne, r_8), (anne, r_{10})\}$.
After running the users roles merging algorithm, the constructed eleven roles will be merged into three roles only, and the user role assignment set will end up having three pairs as shown in the following: $R = \{ r_a \equiv r_1, r_3, r_7, r_9, r_{11}, r_b \equiv r_2, r_4, r_8, r_{10}, r_c \equiv r_5, r_6\}$.
$UA = \{(alex, r_c), (bob, r_a), (anne, r_b)\}$.

Algorithm 2 Users Roles Merging Algorithm

Require: R: The set of roles
Require: $U(r)$: Returns the set of users assigned to the role r.
Require: $RP(r)$: Returns the set of role pairs associated with the role r. .

for each $r_i, r_j \in R$ **do**
  if $U(r_i) = U(r_j) \wedge r_i \neq r_j$ **then**
    for each $rp_k \in RP(r_i)$ **do**
      $rp_k.r = r_j$
    end for
    $R = R \setminus r_i$
      ▷ Delete all UA pairs related to r_i
    for each $(a, b) \in UA$ **do**
      if $b = r_i$ **then**
        $UA = UA \setminus (a, b)$
      end if
    end for
  end if
end for

5.5 The output of EGRBAC Constructing Approach on HABAC Use Case

The output of EGRBAC role constructing algorithm for the Use case in Table 2 is shown in Table 7. Maximum number of created device roles is $O(|OPA| + |DA|)$. Since we create an environment role and an environment condition for each logical environment condition, maximum number of environment roles and conditions is $\Omega(|ESA|)$. Finally, maximum number of user roles is $O(2^{|SA|+|ESA|})$.

6 Discussion and Future Directions

Our proposed HABAC model is a user to device access control model. It captures different users, environment, operations, and devices characteristics. Therefore, it is a dynamic model. It is a fine grained model; since it is capable of giving users access to some operations within a single device without the need to give them access to the entire device. Our approach of constructing HABAC from EGRBAC is a simple, straightforward approach that is capable of translating EGRBAC configuration into an HABAC policy configuration. However, as we discussed in Section 4, in HABAC we can not create something equivalent to EGRBAC *PRConstraints*. This makes it troublesome to prevent future authorization of specific users to access specific operations on specific devices, since the only way to do so is dynamically at enforcement time when the user is trying to access the prohibited operation, unlike EGRBAC in which we can enforce this prevention at assignment time. In EGRBAC, determining the role structure could take a lot of efforts, but when completed it is easy to define who has what permissions, and who is not allowed to have a future access to specific permissions. On the other hand, in HABAC this is not achievable. In addition to users, sessions, devices, and operations static attributes, our EGRBAC constructing approach is capable of handling HABAC policies that contain environment attributes. Due to some limitations in EGRBAC, our approach can't handle HABAC policies that involve users, devices and operations dynamic attributes. As explained in Section 3, dynamic attributes are those attributes that are rapidly changing without involving administration actions. For instance device temperature. In our approach, we translate device, and permission attributes instances into device roles. In EGRBAC, device roles are means of categorizing permissions of different devices according to relatively static characteristics. when a permission is assigned to a specific device role, then it is part of that device role until some administration change happens, there is no way to dynamically activates and deactivates neither device roles, nor assignment of permissions to different device roles. For example, in HABAC we may have a device attribute $device\_temperature : d : D \longleftarrow \{Low, High\}$. We can easily configure an access policy that authorizes some users to access a device d_x only if $device\_temperature(d_x) = Low$. To do so in EGRBAC we have two options, the first one is by creating two device roles one for high temperature, and another for low

Table 7: The output of EGRBAC Constructing Approach on HABAC Use Case

(a) $U_{EGRBAC} = U_{H-ABAC}, D_{EGRBAC} = D_{H-ABAC}, OP_{EGRBAC} = OP_{H-ABAC}, P_{EGRBAC} = \{(TV, G), (TV, PG), (PlayStation, A3), (PlayStation, A7), (PlayStation, A12), (PlayStation, BuyGames), (Oven, ON), (Oven, OFF), (Fridge, Open), (Fridge, Close), (FrontDoor, Lock), (FrontDoor, Unlock)\}$
(b) $DR = \{DangerouseKitchenDevices = True, DangerouseKitchenDevices = False, KidsFriendly = True, KidsFriendly = False, RemPerm\}$.
(c) $PDRA = \{((TV, G), KidsFriendly = True), (PlayStation, A3), KidsFriendly = True), ((PlayStation, A7), KidsFriendly = True), ((TV, PG), KidsFriendly = False), ((PlayStation, A12), KidsFriendly = False), ((PlayStation, BuyGames), KidsFriendly = False), ((Oven, ON), DangerouseKitchenDevices = True), ((Oven, OFF), DangerouseKitchenDevices = True), ((Fridge, Open), DangerouseKitchenDevices = False), ((Fridge, Close), DangerouseKitchenDevices = False), ((FrontDoor, Lock), RemPerm), ((FrontDoor, Unlock), RemPerm)\}$.
(d) $EC = \{True, ec_1 \equiv ParentInKitchen(current) = True, ec_2 \equiv day(current) \in \{Sa, S\}, ec_3 \equiv 12:00 \leq time(current) \leq 19:00, ec_4 \equiv day(current) \in \{M, T, W, Th, F\}, ec_5 \equiv 17:00 \leq time(current) \leq 19:00\}$.
(e) $ER = \{Any\_Time, er_1 \equiv ParentInKitchen(current) = True, er_2 \equiv day(current) \in \{Sa, S\} \equiv Weekend, er_3 \equiv 12:00 \leq time(current) \leq 19:00 \equiv Afternoon\ and\ Evening, er_4 \equiv day(current) \in \{M, T, W, Th, F\} \equiv Weekdays, er_5 \equiv 17:00 \leq time(current) \leq 19:00 \equiv Evening\}$.
(f) $EA = \{(\{True\}, Any\_Time), (\{ec_1\}, er_1), (\{ec_2\}, er_2), (\{ec_3\}, er_3), (\{ec_4\}, er_4), (\{ec_5\}, er_5)\}$.
(g) $R = \{r_a, r_b, r_c\}$.
(h) $UA = \{(bob, r_a), (anne, r_b), (alex, r_c)\}$.
(i) $RP = \{(r_a, Any\_Time), (r_b, Any\_Time), (r_b, \{er_1\}), (r_c, \{er_2, er_3\}), (r_c, \{er_4, er_5\})\}$.
(j) $RPDRA = \{((r_a, Any_Time), DangerouseKitchenDevices = True), ((r_a, Any\_Time), DangerouseKitchenDevices = False), ((r_a, Any\_Time), KidsFriendly = True), ((r_a, Any\_Time), KidsFriendly = False), ((r_a, Any\_Time), RemPerm), ((r_b, \{er_1\}), DangerouseKitchenDevices = True), ((r_b, Any\_Time), DangerouseKitchenDevices = False), ((r_b, Any\_Time), KidsFriendly = True), ((r_b, Any\_Time), KidsFriendly = False), ((r_c, \{er_2, er_3\}), KidsFriendly = True), ((r_c, \{er_4, er_5\}), KidsFriendly = True)\}$.

temperature for each device. This will result in a role explosion. Furthermore, there is no mechanism in EGRBAC that can dynamically activates d_x's high temperature device role while deactivating the low temperature device role when the temperature of dr_x is high and vice versa. The second option is for those devices which have similar access conditions we create a device role for low temperature, and a device role for high temperature. However, there is no way to dynamically activates or deactivates devices membership in different device roles according to their temperatures. The similar argument holds when we deal with dynamic user attributes. Our EGRBAC constructing approach didn't consider the following: (1)Policies that compare two different types of attributes. (2)HABAC configurations that involve user attributes constraints and session attributes constraints. Unfortunately, we don't have enough space to analyze and include such use cases. However, this may be a possible future direction.

From the above, a hybrid model combining HABAC and EGRBAC features may be the most suitable for smart home IoT, and likely more generally. A possible future direction is developing a combined model which prevents a "role explosion" to cover different authorizations for every possible user, environment, operation, and device conditions while retains advantages of EGRBAC, such as ease of user provisioning and the ability to specify the maximum permissions available to each user.

7 Conclusion

In this paper, we introduce HABAC access control model for smart home IoT. It is a dynamic, fine grained ABAC based model that captures different attributes of users, environment, operations, and devices. We provide a use case scenario demonstration. Moreover, we compare the theoretical expressive power of our model to EGRBAC [4] which is a dynamic contextual aware RBAC based access control model. We do that by providing approaches for converting an HABAC specification to EGRBAC and vice versa. We found that while EGRBAC is capable of handling environment attributes, and relatively static users, and device attributes, it is incapable of handling relatively dynamic users, and devices attributes. On the other hand, unlike EGRBAC, in HABAC it is hard to prevent future authorization of specific users to access specific operations on specific devices. In conclusion, we believe that a hybrid model retaining HABAC and EGRBAC features may be the most suitable for smart home IoT, and likely more generally.

Acknowledgement

This work is partially supported by NSF CREST Grant 1736209.

References

[1] [n.d.]. Internet of things. https://en.wikipedia.org/wiki/Internet_of_things.
[2] G. Ali, et al. 2019. Blockchain based permission delegation and access control in Internet of Things (BACI). *Computers & Security* (2019).
[3] M. Alramadhan and K. Sha. 2017. An overview of access control mechanisms for internet of things. In *ICCCN*. IEEE.
[4] S. Ameer, et al. 2020. The EGRBAC Model for Smart Home IoT. In *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)*. IEEE.
[5] O. Arias, et al. 2015. Privacy and security in internet of things and wearable devices. *TMSCS* (2015).
[6] S. Bandara, et al. 2016. Access control framework for api-enabled devices in smart buildings. In *APCC*. IEEE.
[7] E. Barka, et al. 2015. Securing the web of things with role-based access control. In *C2SI*. Springer.
[8] B. Bezawada, et al. 2018. Securing Home IoT Environments with Attribute-Based Access Control. In *ABAC'18*. ACM.
[9] S. Bhatt, et al. 2017. Access control model for AWS internet of things. In *International Conference on Network and System Security*.
[10] S. Bhatt and R. Sandhu. 2020. ABAC-CC: Attribute-Based Access Control and Communication Control for Internet of Things. In *Proceedings of the 25th ACM Symposium on Access Control Models and Technologies*.
[11] M. J. Covington, et al. 2000. *Generalized role-based access control for securing future applications*. Technical Report. Georgia Tech.
[12] T. Denning, et al. 2013. Computer security and the modern home. *Commun. ACM* (2013).
[13] S. Ding, et al. 2019. A novel attribute-based access control scheme using blockchain for IoT. *IEEE Access* (2019).
[14] E. Fernandes, et al. 2016. Security analysis of emerging smart home applications. In *SP*. IEEE.
[15] E. Fernandes, et al. [n.d.]. Flowfence: Practical data protection for emerging iot application frameworks. In *USENIX Security 16*.
[16] D. F. Ferraiolo, et al. 2001. Proposed NIST standard for role-based access control. *TISSEC* (2001).
[17] J. Granjal, et al. 2015. Security for the internet of things: a survey of existing protocols and open research issues. *IEEE Comm. Surv. & Tutorials* (2015).
[18] Z. Guoping and G. Wentao. 2011. The research of access control based on UCON in the internet of things. *Journal of Software* (2011).
[19] M. Gupta, et al. 2019. Dynamic groups and attribute-based access control for next-generation smart cars. In *Ninth ACM Conference on Data and Application Security and Privacy*.
[20] M. Gupta and R. Sandhu. 2018. Authorization framework for secure cloud assisted connected cars and vehicular internet of things. In *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*.
[21] W. He, et al. 2018. Rethinking access control and authentication for the home internet of things (IoT). In *USENIX Security 18*.
[22] K. Hill. 2013. Baby Monitor Hack Could Happen To 40,000 Other Foscam Users. https://www.forbes.com/sites/kashmirhill/2013/08/27/baby-monitor-hack-could-happen-to-40000-other-foscam-users/613ec55458b5.
[23] G. Ho, et al. 2016. Smart locks: Lessons for securing commodity internet of things devices. In *ASIA CCS '16*. ACM.
[24] V. C. Hu, et al. 2015. Attribute-based access control. *Comp.* (2015).
[25] X. Jin, et al. 2012. A unified attribute-based access control model covering DAC, MAC and RBAC. In *IFIP Annual Conf. on Data and App. Sec.*
[26] J. Jindou, et al. 2012. Access control method for web of things based on role and sns. In *CIT 2012*. IEEE.
[27] S. Kaiwen and Y. Lihua. 2014. Attribute-role-based hybrid access control in the internet of things. In *APWeb*. Springer.
[28] F. Martinelli, et al. 2018. Too long, did not enforce: a qualitative hierarchical risk-aware data usage control model for complex policies in distributed environments. In *CPSS '18*. ACM.
[29] B. Mitra, et al. 2016. A survey of role mining. *Comput. Surveys* (2016).
[30] A. Mutsvangwa, et al. 2016. Secured access control architecture consideration for smart grids. In *IEEE PES PowerAfrica*.
[31] O. Novo. 2018. Blockchain meets IoT: An architecture for scalable access management in IoT. *IEEE IoT Journal* (2018).
[32] A. Ouaddah, et al. 2017. Towards a novel privacy-preserving access control model based on blockchain technology in IoT. In *Europe and MENA Coop. Adv. in Inf. and Comm. Tech.* Springer.
[33] A. Ouaddah, et al. 2017. Access control in the Internet of Things: Big challenges and new opportunities. *Comp. NW* 112 (2017).
[34] J. Park. 2003. *Usage control: A unified framework for next generation access control*. Ph.D. Dissertation. George Mason University.
[35] J. Park and R. Sandhu. 2002. Towards usage control models: beyond traditional access control. In *SACMAT '02*. ACM.
[36] J. Qiu, et al. 2020. A survey on access control in the age of internet of things. *IEEE Internet of Things Journal* (2020).
[37] S. Ravidas, et al. 2019. Access control in Internet-of-Things: A survey. *Journal of Network and Computer Applications* (2019).
[38] R. Sandhu. 1998. Role-based access control. In *Advances in computers*. Vol. 46.
[39] A. Tilley. 2016. How A Few Words to Apple's Siri Unlocked a Man's Front Door. http://www.forbes.com/sites/aarontilley/2016/09/21/apple-homekit-siri-security.
[40] M. Tripunitara and N. Li. 2007. A theory for comparing the expressive power of access control models. *Journal of Computer Security* 15 (02 2007), 231–272.
[41] B. Ur, et al. 2013. The current state of access control for smart devices in homes. In *HUPS*.
[42] Y. Xie, et al. 2015. Three-layers secure access control for cloud-based smart grids. In *IEEE 82nd VTC2015-Fall*. IEEE.
[43] N. Ye, et al. 2014. An efficient authentication and access control scheme for perception layer of internet of things. *Applied Math. & Inf. Sciences* (2014).
[44] G. Zhang and J. Tian. 2010. An extended role based access control model for the Internet of Things. In *2010 ICINA*. IEEE.