2022 IEEE 4th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA) | 978-1-6654-7408-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/TPS-ISA56441.2022.00035

Scenario-Driven Device-to-Device Access Control in Smart Home IoT

Mehrnoosh Shakarami*, James Benson, Ravi Sandhu
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, San Antonio, US
mehrnoosh.shakarami@my.utsa.edu, james.benson@utsa.edu, ravi.sandhu@utsa.edu

***Abstract*—The Internet of Things (IoT) has been widely integrated in people's everyday lives. As an infrastructure of connected heterogeneous devices, IoT has not yet achieved the seamless integration of device-to-device collaboration which is necessary for real-life home automation. Smart home IoT devices expect to exchange their collected data or status in certain circumstances, in spite of their heterogeneity, viz. working with different communication protocols, IoT platforms, middleware, data and semantics. Deploying appropriate access control models and mechanisms is of utmost importance as any unauthorized access to data could have a cascading violation of privacy, safety and security of users. In this work, we propose a novel device-to-device access control paradigm in the smart home IoT. Our approach relies on message passing as the paradigm for device-to-device interactions. We further introduce *actions* and *scenarios* reflecting the chain of events in the smart home context, which facilitates *scenario-driven attribute-based access control*. Each scenario is triggered by *triggering events*, based on previously set administrative definitions. We define totally ordered sets of triggering events using *priorities* to enable conflict resolution for devices which may run into conflicting commands delivered though messages in different ongoing scenarios. The viability of the proposed approach is substantiated via a formal model and an enforcement architecture, backed up by a proof-of-concept implementation which affirms a trade-off between required authorization and efficacy. Potential future challenges are explored in the context of smart home IoT platforms.**

***Index Terms*—device-to-device access control, Attribute-Based Access Control, message passing, scenario-driven access control, smart home IoT**

I. Introduction

Smart IoT environment in which device-to-device (hereafter D2D) communications happen seamlessly and directly among heterogeneous devices, is currently more of a vision than reality. Considering home IoT devices as an ecosystem with inter-communications provides a holistic perspective toward home automation and brings added convenience. However, it introduces potential risks to the safety and privacy of home users. For instance, an attacker may compromise a device to misuse it as a breaking point for unauthorized access to the network to which the device is attached [1], [2]. Although there are handful of studies on individual parts of the smart home IoT, e.g., device authentication [3]–[5], communication protocols [6]–[8], and home automation applications [9]–[11], research body on security of *interactions* in smart home is quite scarce. In this context, securing D2D interaction through authorization of the flow of information among devices is pivotal and could be mediated via access control approaches.

Although there is a research body on regulating user-to-device (hereafter U2D) access in smart homes, no previous work has been devised to regulate the device-to-device communications through specification of an access control model. Our main contribution is to formulate an access control model which governs authorized flow of information between home IoT devices using Attribute-Based Access Control (ABAC), which is the first attempt in this context to the best of our knowledge. An ABAC policy enables capturing the dynamics and fluidity of the smart home environment by entailing contextual/environmental attributes such as time and location. It also facilitates defining authorization rules based on attributes of involved devices in D2D communications.

Another contribution in this paper is utilizing the message passing paradigm for D2D access control. In its essence, in our approach human administrators, i.e., homeowners or parents, define a set of authorized message flows between devices through establishing access control rules, which is defined based on sender/receiver device attributes, as well as contextual attributes. We consider such tasks to be administrative, which precede the regulation of D2D access control at operational level. Thereafter, no user intervention is required for mediation of D2D access in our approach.

Further contribution is extending our model to include scenarios as a set of actions that may occur in the home IoT environment. Different sets of actions in the smart home IoT could be initiated by a trigger, i.e., an event or a set of events in the smart home. As any D2D communication is considered to be done via message passing, we conclude each trigger initiates a set of message passing among devices in the smart home IoT environment. Then actions would be coupled with their triggering events to define scenarios. By defining priorities between triggers, we equip our model with conflict resolution. So, if a device is involved in conflicting ongoing scenarios, that could be resolved without human intervention.

We acknowledge the complications of the evolving D2D communication, i.e., different platforms, communication protocols and data models, and accordingly design a consolidated access control approach in this context. We consider direct communication of heterogeneous IoT devices out of scope [12] and focus our work to serve as a gateway-enabled initiative

*First Author works at Amazon now. This work was done prior to joining Amazon.

978-1-6654-7408-5/22/$31.00 ©2022 IEEE
DOI 10.1109/TPS-ISA56441.2022.00035

in the scarcely investigated area of D2D access control. The rest of this paper is organized as follows. Section II reviews the related work. Proposed message-based ABAC model, the threat model, and a smart home use case are discussed in Section III. Our scenario-based access control proposal along with a smart home use case is presented in Section IV. Section V includes our enforcement architecture. The experimental setup and results are discussed in Section VI. Section VII discusses the properties of the proposed approach and future directions of research. Section VIII concludes the paper.

II. Related Work

We discuss related works in three categories: D2D communication, message-passing in IoT environments, and access control solutions for smart home IoT.

A. Device-to-Device Communication

Seamless interoperability among IoT devices is imperative for ongoing evolution of the IoT ecosystem. When co-located devices that want to interoperate use heterogeneous communication technologies it becomes a challenging issue [13], [14]. Multiple attempts including IEEE P2413 [15], OneM2M[1], BigIoT[2], Agile[3], SymbIoTec [16], and BIoTope [17] try to address the fragmentation of IoT application/services by designing reference architectures, technical specifications, unified web APIs or providing abstractions or standard libraries. Resource sharing and discovery for rapid cross-platform applications is another area of investigation [18]. In another effort, 3rd-generation partnership project (3GPP)[4] tries to provide cellular IoT, and is utilized in some research works [19], [20].

Similarly, semantic models based on OSGi framework[5] seek network-level interoperability in home environments [21]–[23]. Some efforts rely on social network to build a device/standard agnostic platform for heterogeneous IoT connection [24]–[26]. Despite many ongoing efforts there is no de-facto standard and there will be none in foreseeable future [27]. INTERIoT [28] is another approach which is aimed not for providing a reference nor a standard model, rather its goal is seamless cooperation/integration of heterogeneous IoT platforms based on a layered approach. Heterogeneity in all levels of IoT technology, including device, networking, middleware, application and data/semantics of IoT scenarios, makes heterogeneous IoT devices communications cumbersome. Some enabling technologies and their challenges are discussed in [29]. Interestingly, access control modeling is left out of consideration in all these efforts.

B. Access Control in Smart Home IoT

Unauthorized access to the information flow in any IoT environment may impose critical safety and privacy issues to IoT users [30]–[35]. As a popular IoT application, smart home IoT demands for specific access control models tailored for its especial requirements [36]–[40]. Proposed access control models for smart home IoT, surprisingly, are not many. Some researchers rely on Role-Based Access Control (RBAC) [41], as the easiest for home users to adopt [42] and manage [38], [43]. Capability-Based Access Control (CapBAC) [44] and Attribute-Based Access Control (ABAC) [45]–[47] are also utilized to propose authorization solutions for resource-constrained IoT deployments. Utilizing blockchain for smart home access control is widely investigated, yet without proposing a formal access control model [48]–[52].

Bezawada et al. [53] is one of the few works which provided some use cases along with their NIST NGAC[6] enforcement architecture considering D2D communications, however, no general access control model/policy is defined. Ruledger [54] is a ledger-based framework which considers D2D interactions, but the goal is to ensure rule integrity in trigger-action IoT platforms, not access control. Some research works considered heterogeneity of IoT devices, even so their focus is on U2D interactions over heterogeneous IoT platforms [55]. A capability-based access delegation approach for D2D interaction has been proposed in [56]. However, it contains no D2D access control model.

C. Message-Based Communication in IoT Environments

Some proposals in IoT environment utilize messages for control and communication [24], authentication [5], discovery and configuration of new IoT devices [57], and routing [58]. There are also some research works on securing communications using access control models [59]. Alshehri and Sandhu [60] identified the need for controlling data and communication in IoT environments. A general conceptual model for attribute-based communication control has also been developed [61]. Our proposal is the first access control model based on the message passing paradigm for authorization of device-to-device communications.

III. Message-Based D2D ABAC Authorization Model

In order to design an appropriate access control framework for any IoT environment capturing the context which thereby incorporates the dynamicity of the IoT ecosystem is required [31]. The risk of insecurity due to unauthorized access varies by the context in which the IoT device operates [1]. The context of an entity could be interpreted as any information to characterize its situation [62]. For an IoT device, the context could include its location, battery status or owner. There are many access control proposals in IoT environments which design or create the IoT authorization policies while capturing the context relying on semantic web technologies [45], RBAC extensions [63], ABAC [64], OrBAC [65] or combination of different access control paradigms [66].

[1]https://www.onem2m.org/
[2]https://cordis.europa.eu/project/id/688038
[3]http://agile-iot.eu/about/
[4]https://www.3gpp.org/news-events/1906-c_iot
[5]http://docs.osgi.org/download/r4v42/r4.core.pdf
[6]https://www.nist.gov/topics/identity-access-management/policy-machine-and-next-generation-access-control

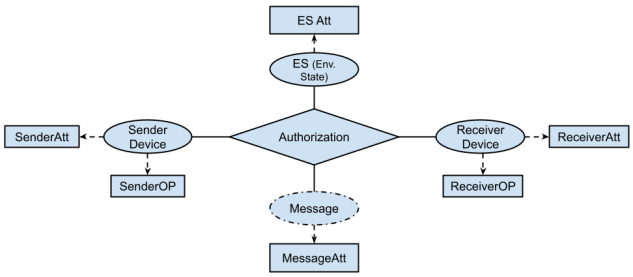

Fig. 1. Device-to-Device ABAC Model

In this paper, we capture the IoT device context via the contents of communicated messages in device-to-device interactions. In our approach, a message is a distinct structured entity being communicated between two IoT devices. Message attributes are derived from its content. The contents of a message are structured in a *(key,value)* format and reflect the communication context, including the purpose of communication which is reflected in the message classification followed by its corresponding attributes in the message. Unique to our approach is a message passing paradigm for IoT device-to-device *access control*. No two IoT devices would be able to communicate with each other, unless there is a corresponding authorization rule defining their possible type of communication. In other words, the set of authorized device-to-device communications is defined through establishing access control rules based on attributes of environment, sender and receiver devices, and the transmitted message. Notably, for the first time our model presents the IoT device-to-device communication to be without human intervention.

A. Conceptual Model

Our conceptual ABAC model is depicted in Fig. 1. There are two IoT device endpoints in each communication, subject to authorization, denoted as *sender device* and *receiver device*. Each of these two communicating endpoints has its own assigned attributes, such as ID, type, location, etc. The attributes of sender/receiver devices are represented as *SenderAtt*/*ReceiverAtt* respectively. Moreover, for each endpoint device there is a set of available functionalities, defined by its manufacturer, and represented as *SenderOP*/*ReceiverOP* as depicted in Fig. 1. Message is introduced in our model as a new element of D2D access control. Message attributes are extracted from its content and present the context of communication, i.e., the attributes of the environment (smart home) in which messages are transmitted. Message attributes are represented as *MessageAtt* in Fig. 1. As messages are transitory elements of the system, which do not exist until being sent/received by corresponding devices, we depict the message entity in a dotted circle in Fig. 1.

To be in accord with the highly dynamic nature of smart home IoT, we consider environment state as an entity in our model, represented as *ES* and utilize its associated attributes, a.k.a. *ESAtt* for access control decisions. Examples of *ESAtt* include daylight, time of the day, weather conditions, etc. *Authorization* function is defined based on different entities' attributes as depicted in Fig. 1. When one IoT device requests to communicate with another IoT device, the authorization function, which has been defined by *CheckAccess* predicate in our formal representation, allows or denies the access request.

B. Formal Model

Our model's formalization is presented in Table I. The basic components of the model are discussed below.

Core Components. Include the set of IoT devices (D), and their available operations (OP) which are vendor-defined through Device operation assignment (DOA) relation. The environment state (ES) is the set of environment attributes representing. D, ES and M (set of messages) are considered as entities (Ent) in our model.

Attribute Functions. We consider each entity's attribute value type, represented as $AttValueType$ to be either *atomic*, i.e. it would have one of the values of its range at each moment, or a set, which could be assigned to a subset from its range. Moreover, each device attribute is either *static*, i.e. its value is fixed and statically assigned by its owner/manufacturer, or *dynamic*, when its value assignment could change as the side effect of communications with other devices or the operations done by the device itself. The assignment type for each attribute is represented as $AttAssignType$ relation. EAA and DAA denote the environment/device attribute assignment relations which associate attributes to the environment state and device entities respectively.

Message and Message Functions. M is a set of messages. Each message is a set of *n* attributes in the form of *(key,value)* pairs, communicated between two devices. Multi-messaging and message broadcasts are out of scope. As messages are transient entities, its sender's ID and receiver's ID are not contained in the template of the message definition. The message type is defined as a mandatory first attribute of every message, which determines the rest of required attributes to be included in the message. Every message belongs to one of the classifications indicated in *typeSet*, restricted to *"query"*, *"command"*, and *"info"*. Message type is reflected in the first pair of its attribute represented as $("type", value_1)$ in which $value_1$ could be one of the predefined types in *typeSet*.

A message of type *"query"*, inquires into the value of attributes of the receiver, for instance the outdoor camera may send a *query* message to the door lock to know its "locked" attribute value. This message might be frequently communicated, to monitor any changes in the attribute's value. The frequency of communication depends on the device, its state changes, and the commands issued. A message with type *"command"* orders the receiver device to perform an operation towards its environment, e.g., a security camera may command the door lock to be locked. An *"info"* message informs the receiver about the values of some of the attributes of the sender. Any message of *"query"* or *"command"* requires receiving back a message of type *"info"* in response, which provides the requested attribute value or acknowledges the command's fulfillment. An *"info"* message needs no response.

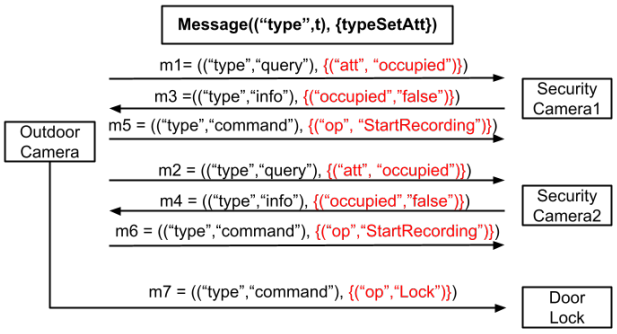

Fig. 2. Smart Home Use Case for Device-to-Device Communication

Check Access Predicate. This function would be evaluated for each message communication to TRUE or FALSE which respectively indicates allowance or denial of specified communication in the message, from sender to receiver. In order to authorize a message communication between a sender device and a receiver device, two functions would be checked by *CheckAccess* function. *CheckAtt* checks the *feasibility* of the message which is intended to be communicated, that is determined based on message type. If a message is of type *”query”*, the requested attributes by sender must be a subset of attributes defined for the receiver device through *DAA* relation. Similarly, for the *”info”* messages, the communicated set of attributes has be a subset of sender device's attributes, defined via *DAA*.

For messages of type *”command”*, the operation which senders ask the receiver to do must be a functionality which is available to the receiver and defined through *DOA* by its vendor/manufacturer. For instance, if a sender device asks a receiver's device to do an operation which is not defined for the receiver, this function returns ”false”.

C. Smart Home Use Case

This use case represents how the components of our model should be configured to enforce D2D access control in a smart home IoT environment without any need for human intervention. Consider a single owner (say Alice) home, which comprises a smart door lock, two indoor smart security cameras and a smart outdoor camera. We assume all indoor and outdoor cameras are equipped with presence sensors, so the outdoor camera can detect Alice's arrival/departure and indoor security cameras could detect whether anybody is home.

Fig. 2 represents our use case in which the messages are shown without *sender* and *receiver* fields for the sake of simplicity. The outdoor camera takes the initiative when it detects Alice is leaving. It then checks the *occupied* attribute of security cameras via sending *query* messages. If the house indicated to be vacant (*occupied* returned as FALSE by all security cameras), then outdoor camera sends a *command* message to all cameras to *startRecording*, and a *command* message to the door lock to be *locked*. As soon as Alice is detected to be back by the outdoor camera, it sends an *info* message to involved devices, so the recording would be stopped, and the door unlocked. Our proposed access control model could be configured as shown in Table II to implement this use case. Notably, *Check Access Predicate* depicts the propositions which are defined to authorize message communications among devices. If any device asks for any attribute of another device, which has not been authorized via the check access predicate, that communication would be denied.

TABLE I
MESSAGE-BASED ABAC MODEL FORMALIZATION

Core Components

$-D$ is a set of smart home IoT devices deployed by homeowner.
$-OP$ is a set of operations available on different devices in the system (manufacturer specified).
$-ES = \{current\}$ is the singleton set, representing the environment state at the current time instant.
$-Ent = D \cup ES \cup M$ is the set of entities in the system, where the set of messages M is defined below.
$-DOA : D \rightarrow 2^{OP}$ is a one to many relation which associates a device to its available operations as specified by the device manufacturer.

Attribute Functions

$-DAA, EAA$ are respectively sets of attribute functions which associate a device or the current environment state with attribute values.
$-attValueType : DAA \cup EAA \rightarrow \{atomic, set\}$
$-\forall att \in DAA \cup EAA,\ Range(att)$, is the attribute range, a finite set of atomic values.
−Each $att \in DAA \cup EAA$ maps a device/environment to a single *atomic* value or to a finite *set* of values, as follows:
$-att : DAA \cup EAA \rightarrow$

$$\begin{cases} Range(att) & \text{if } attValueType(att) = atomic \\ 2^{Range(att)} & \text{if } attValueType(att) = set \end{cases}$$

$-attAssignType : DAA \cup EAA \rightarrow$

$$\begin{cases} static & \text{set/changed via administrative actions} \\ dynamic & \text{set/changed automatically by deployed sensors in home} \end{cases}$$

Message and Message Functions

$-M = \{m\}$ is the set of all messages in the system.
$-m = \{(att_1, value_1), (att_2, value_2), ..., (att_n, value_n)\}$, represents any single message in the system with n different attributes, each of which is indicated as a $(key, value)$ pair.
$-typeSet = \{”query”, ”command”, ”info”\}$ is a mandatory first attribute in every message which indicates its *type* and thereby the rest of message attributes.
−For each $m \in M$, we assume the first attribute determines the type of the message: $att_1 \in typeSet$
$-typeSetAtt : M \rightarrow 2^{DAA} \cup 2^{DOA}$, is a function which indicates the set of attribute keys required to be communicated based on the message type, supposed to be communicated via $\{att_2, ..., att_n\}$ in each message.

Check Access Predicate

$-CheckAccess$ is evaluated when a sender device (s) tries to send a message (m) to a receiver device (r) in context of current environment state ($current$) and is evaluated based on following formula:
$-CheckAccess(s : D, m : M, r : D, current : ES) \equiv$
$CheckAtt(s : D, m : M, r : D, current : ES) \quad \wedge$
$Authorization(s : D, m : M, r : D, current : ES)$
$-CheckAtt = True \iff typeSetAtt(m) =$

$$\begin{cases} \subseteq 2^{DAA(r)} & \text{if } m.value_1 = ”query” \\ \in DOA(r) & \text{if } m.value_1 = ”command” \\ \subseteq 2^{DAA(s)} & \text{if } m.value_1 = ”info” \end{cases}$$

$-Authorization(s : D, m : M, r : D, current : ES)$ is a logical proposition which could be evaluated to either True or False and is created using following policy rules.
$-p \equiv (p) \mid \neg p \mid p \wedge p \mid p \vee p \mid \exists x \in set.p \mid \forall x \in set.p \mid set\ \Delta\ set \mid atomic \in set$
$-\Delta \equiv \subset | \subseteq | \not\subseteq | \cap | \cup$

TABLE II
MESSAGE-BASED D2D USE CASE SCENARIO

Core Components
D = {OutdoorCamera, SecurityCamera$_1$, SecurityCamera$_2$, DoorLock}
OP = {Lock, Unlock, StartRecording, StopRecording, TurnOn, TurnOff}
ES = {current}
DOA = {(SecurityCamera$_1$,{StartRecording, StopRecording}), (SecurityCamera$_2$,{StartRecording, StopRecording}),(OutdoorCamera, {StartRecording, StopRecording}), (DoorLock,{Lock, Unlock})}
DAA = {(SecurityCamera$_1$,{id, type, location, recording, occupied}), (SecurityCamera$_2$,{id, type, location, recording, occupied}), (OutdoorCamera, {id, type, location, recording, incident}), (DoorLock, {id, type, location, locked})}
EAA = {day, time}

Attribute Functions
id(SecurityCamera$_1$)="sc1", type(SecurityCamera$_1$)= "camera", location(SecurityCamera$_1$)= "indoor", recording(SecurityCamera$_1$) = {"true","false"}, occupied(SecurityCamera$_1$) = {"true","false"}
id(SecurityCamera$_2$)="sc2", type(SecurityCamera$_1$)= "camera", location(SecurityCamera$_1$)= "indoor", recording(SecurityCamera$_2$) = {"true","false"}, occupied(SecurityCamera$_2$) = {"true","false"} id(OutdoorCamera) = "oc1",
type(SecurityCamera$_1$)= "camera", location(SecurityCamera$_1$)= "outdoor", recording(OutdoorCamera$_1$) = {"true","false"} incident(OutdoorCamera$_1$) = {"coming","leaving"} id(DoorLock) = "dl1", locked(DoorLock) = {"true", "false"}, type(DoorLock)= "lock", location(SecurityCamera$_1$)= "mainEntrance"

$$\begin{cases} \text{attAssignType = static,} & \text{if att}\in\{"id","type","location"\} \\ \text{attAssignType = dynamic,} & \text{otherwise} \end{cases}$$

Message and Message Functions
$M = \{m_1, m_2, m_3, m_4, m_5, m_6, m_7\}$
$m_1 = \{("type","push"),\{("att","occupied")\}\}$, $m_2 = \{("type","push"),\{("att","occupied")\}\}$,
$m_3 = \{("type","push"),\{("occupied","false")\}\}$, $m_4 = \{("type","push"),\{("occupied","false")\}\}$,
$m_5 = \{("type","com"),\{("op","StartRecording")\}\}$, $m_6 = \{("type","com"),\{("op","StartRecording")\}\}$,
$m_7 = \{("type","com"),\{("op","Lock")\}\}$

Check Access Predicate
$-CheckAccess(s:D, m:M, r:D, current:ES) \equiv$
$CheckAtt(s:D, m:M, r:D, current:ES) \wedge Authorization(s:D, m:M, r:D, current:ES)$

$$-CheckAtt = True \iff typeSetAtt(m) = \begin{cases} \subseteq 2^{DAA(r)} & \text{if } m.value_1 = "query" \\ \in DOA(r) & \text{if } m.value_1 = "command" \\ \subseteq 2^{DAA(s)} & \text{if } m.value_1 = "info" \end{cases}$$

$-Authorization(s:D, m:M, r:D, current:ES) \equiv q_1 \vee q_2 \vee q_3 \vee q_4$

$$q_1 = \Big[\Big(m.att_1 = "query"\Big) \wedge \Big(typeSetAtt(m) \in \{"recording","occupied"\}\Big) \wedge \Big(type(s) = type(r) = "cameras"\Big) \wedge \Big(location(s) = "outdoor"\Big) \wedge \Big(location(r) = "indoor"\Big)\Big]$$

$$q_2 = \Big[\Big(m.att_1 = "info"\Big) \wedge \Big(typeSetAtt(m) \in \{"recording","occupied"\}\Big) \wedge \Big(type(s) = type(r) = "cameras"\Big) \wedge \Big(location(s) = "indoor"\Big) \wedge \Big(location(r) = "outdoor"\Big)\Big]$$

$$q_3 = \Big[\Big(m.att_1 = "command"\Big) \wedge \Big(typeSetAtt(m) \in \{"StartRecording","StopRecording"\}\Big) \wedge \Big(type(s) = type(r) = "cameras"\Big) \wedge \Big(location(s) = "outdoor"\Big) \wedge \Big(location(r) = "indoor"\Big)\Big]$$

$$q_4 = \Big[\Big(m.att_1 = "command"\Big) \wedge \Big(typeSetAtt(m) \in \{"Lock","Unlock"\}\Big) \wedge \Big(type(s) = "cameras"\Big) \wedge \Big(type(r) = "locks"\Big) \wedge \Big(location(s) = "outdoor"\Big) \wedge \Big(location(r) = "mainEntrance"\Big)\Big]$$

D. Threat Model

Smart home IoT devices are adopted by owners to enhance their lives security and convenience. Nevertheless, these devices' susceptibility to cyberattacks could make them disturbing security holes. Some of the possible attack scenarios, a.k.a attack trees, in a smart home environment are discussed in [67], [68]. Everything considered, providing security in complex and dynamic environments such as smart homes remains a noteworthy challenge. We adopt the "Dolev-Yao" (DY) threat model [69] in which communicating endpoints cannot be considered as trusted nodes in the network. According to the DY model, an adversary can tamper with the data through modification, deletion, or insertion of fake information as the communications rely on wireless medium. As confirming evidence of insecure communications, a 2020 IoT threat report announced 98% of all IoT traffic was not encrypted, thus vulnerable to security threats [7].

We are making the following assumptions for the D2D communication in the smart home in our paper:

1) As many IoT devices are not IP-enabled, using a gateway (GW) node in the network is inevitable [12]. We assume the GW node in our model is trustworthy and available, which is a common assumption [70].
2) Attacker is an outsider to the network with the goal of obtaining illegitimate access to available functionalities/operations of smart home IoT devices.
3) We do not consider adversaries to have physical access to IoT devices.

[7]https://app.hushly.com/runtime/content/xVukSNKffbmoOef2

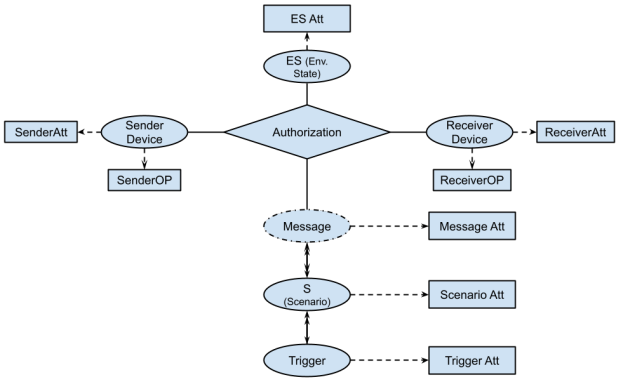

Fig. 3. Device-to-Device Scenario-Driven ABAC Model

Our access control model provides a defense-in-depth prevention/protection against any outsider attack aiming at unauthorized access to an IoT device's operation or information. Our model restricts the set of authorized message communications such that only a subset of one IoT device attribute/operation is accessible to another device. Suppose a well-positioned attacker in the network eavesdrops the communications and impersonates as a legitimate IoT device at home using the known vulnerabilities of the device platform. Our model blocks the attacker's request for arbitrary information s/he may desire to acquire from other home devices. Therefore, it would be arduous for an attacker with no knowledge of established policy rules, which information a device can request from other home devices, yet that information may be of no use to the attacker. Our model is a barrier established, a.k.a a defense-in-depth strategy [8].

IV. Scenario-Based Device-to-Device ABAC Authorization Model

Access decisions in a smart home may conflict due to numerous reasons. Our scenario-driven access control model aims at resolving conflicts which happen because the same device received two messages including conflicting commands.

A. Conceptual Model

Conceptual representation of our model is depicted in Fig. 3. As stated previously, a conflict may happen due to conflicting *command*-type messages received by the same device, initiated by events which we call *triggering events*.

Definition 1: A triggering event is a physical event through an IoT device's state or operation, e.g., the door is opened, or the user is leaving home.

Each trigger initiates a set of actions, which we collectively call an *action*. As we consider any action in smart home IoT to be done via message communication, we define an action as a set of messages communicated among different devices.

Definition 2: An action *a* indicates a message *m* being communicated from a specific sender *s* to a specific receiver *r* and defined as a triplet of $a = (s : D, m : M, r : D)$. *Action* is a set of actions predefined by the administrator/homeowner.

A trigger may initiate a set of actions in the smart home, which we collectively call a *scenario*. So, each scenario is considered as a set of actions done in the smart home environment. A Trigger could provoke one or more scenario(s) in the smart home, i.e. the set of actions which has to be consequently executed in the home. We define priorities as a binary relation among triggers which conceptually reflect the importance of a trigger and its consequences.

Definition 3: priority is a totally ordered set relation, depicted as $(pr, \prec)$ between any two triggering events tr_i and tr_j and is reflected in their (administratively) assigned priority values. So, for any two triggers tr_i and tr_j, it is either $(tr_i \prec tr_j)$ or $(tr_j \prec tr_j)$.

Our conceptual model is represented in Figure 3. Along with other elements of our message-based model, we add extra component *Trigger*, as in Definition 1. Each trigger has attributes shown as *TriggerAtt*, including its administratively assigned priority, the set of device attributes/operations defining that trigger, etc. *Scenario* is also a new component including a set of actions along with its other attributes, shown as *ScenarioAtt*, including its priority, its trigger, and its active status. There is a one-to-many relation between *Scenario* and *Trigger*, which indicates one trigger may initiate multiple scenarios. This relation is defined through TriggeringEvent-Scenario Assignment relation ($TeSA$), as shown in Table III.

Anytime an IoT device sends a message to another IoT device, the Check Access predicate would be evaluated. Check Access includes three functions. CheckAtt which detects the feasibility of communication. CheckPriority examines the message priority and decides if the message should either go through or be disregarded. Authorization defines the set of authorized flow of communications based on attributes of environment, sender and receiver device, and message attributes. Any conflict would be resolved by authorizing the message communication with higher priority.

B. Formal Model

Formal representation of the model is presented in Table III.

Core Components. An extra core component is triggering event set (TE), which could be any change in the IoT device(s)' attributes or an action which has been done by an IoT device. The set of entities include the set of devices, environment state, messages, triggering events and scenarios.

Attribute Functions. $CurrentOP$ and $CurrentPR$ indicate the current operation a device is doing and the priority of the command message currently in-effect, if any. $conflict$ is a set of conflicting functionalities available to one device. Priority is defined as a totally ordered set which defines a strict order between any two triggers in the system. Each trigger would be administratively assigned with its priority, which then at operational level is retrieved via prA function.

Messages, Scenarios and Auxiliary Functions. In this model, A is the set of predefined actions, including a message and its sender and receiver. Each action (a) is representing a

[8] https://csrc.nist.gov/glossary/term/defense_in_depth

TABLE III
SCENARIO-BASED D2D ACCESS CONTROL MODEL

Core Components
$-D$, OP, ES have the same definition as Message-Based Model.
$-DOA$, DAA, EAA remain the same as Message-Based Model.
$-TE \subset D \times \{2^{DOA} \cup 2^{att(d:D)}\}$, is a set of triggering events. att has the same definition as in Table I.
$-Ent = D \cup ES \cup M \cup S \cup TE$ is the set of entities in the system, where the set of messages M and scenarios S are defined below.

Attribute Functions
$-currentOP : D \times \{current\} \rightarrow 2^{DOA} \cup \emptyset$, is the operation each device is doing at the current time instant.
$-currentPR : D \times \{current\} \rightarrow (pr, \prec)$, is the priority of the command that the device is running at the current time instant. $(pr, \prec)$ is defined below.
$-conflict : D \rightarrow 2^{DOA \times DOA} \cup \emptyset$, is a set of conflicting operation pairs defined for each device by the administrator, e.g. homeowner.
$-conflict(d : D) = \{(op_i, op_j) \mid op_i \in DOA(d) \wedge op_j \in DOA(d)\}$.
- All other functions are as defined in Table I.
$-(pr, \prec)$ is a totally ordered set of priorities with at least two distinct elements of $\bot$ and $\top$ which correspondingly represent the lowest and highest priorities in the system.
$-prA : TE \rightarrow (pr, \prec)$ is a function which retrieves priority of the triggering events in the system, originally assigned by system administrator/homeowner.

Messages, Scenarios and Auxiliary Functions
$-A$, is a set of actions in the system.
$-$For each action $a \in A : a = (s : D, m : M, r : D)$, action is defined as a triplet indicating communication of message m from device s to device r.
$-S$ is the set of scenarios in the system defined by system's administrator/homeowner, which is defined below.
$-TeSA : TE \rightarrow 2^S$ is a one-to-many relation which defines a (set) of scenario(s) that would be provoked by a triggering event $te \in TE$.
$-$For each $s \in S : s = \big(Action_s \subseteq A, tr_s : TE, pr_s : (pr, \prec), active : \{"true", "false"\}, id\big)$, and $pr_s = prA(tr_s)$.
$-active(s : S) = \begin{cases} "true" & \text{while } tr_s \text{ is in effect.} \\ "false" & \text{As soon as } tr_s, \text{ triggering event, reverts.} \end{cases}$
$-typeSet = \{"query", "command", "info"\}$ is a mandatory attribute of every message which indicates its *type* and thereby the rest of message attributes.
$-$For each $m \in M$, we assume the first attribute must determine the type of the message and the second attribute must determines its priority: $att_1 \in typeSet$, $att_2 \in (pr, \prec)$
$-typeSetAtt : M \rightarrow 2^{DAA} \cup 2^{DOA}$, is a function which indicates the set of attribute keys required to be communicated based on the message type, supposed to be communicated via $\{att_3, ..., att_n\}$ in each message.
$-msgPrA : D \times M \times D \rightarrow (pr, \prec)$ is a function which is checked each time a device s_d wants to create a command message m and assigns proper priority to it, in order to be sent to device r_d .
$-msgPrA(s : D, m : M, r : D) = \begin{cases} pr_s & \text{if } \Big(m.att_1 = "command"\Big) \wedge \Big(\exists s \in S : active(s) = "true"\Big) \wedge \\ & \quad \Big(\exists a \in Action_s : a.s = s_d \wedge a.r = r_d \wedge typeSetAtt(a.m) = typeSetAtt(m)\Big) \\ \bot & \text{otherwise} \end{cases}$

Chech Access Predicate
$-CheckAccess$ is evaluated when a sender device (s) wants to send a message (m) to a receiver device (r) in the context of current environment state ($current$) and is evaluated based on following formula:
$-CheckAccess(s : D, m : M, r : D, current : ES) \equiv$
$CheckAtt(s : D, m : M, r : D, current : ES) \wedge CheckPriority(s : D, m : M, r : D, current : ES) \wedge$
$Authorization(s : D, m : M, r : D, current : ES)$
$-CheckAtt = True \iff typeSetAtt(m) = \begin{cases} \subseteq 2^{DAA(r)} & \text{if } m.value_1 = "query" \\ \in DOA(r) & \text{if } m.value_1 = "command" \\ \subseteq 2^{DAA(s)} & \text{if } m.value_1 = "info" \end{cases}$
$-CheckPriority(s : D, m : M, r : D, current : ES) \equiv \begin{cases} "false" & \text{if } (m.att_1 = "command") \wedge \\ & \quad \big[((m."op", currentOP(r)) \in conflict(r)) \wedge (m.value_2 \prec currentPR(r)\big] \\ "true" & \text{otherwise} \end{cases}$
$-Authorization(s : D, m : M, r : D, current : ES)$ is a logical proposition which could be evaluated to either True or False and is created using following policy rules.
$-p \equiv (p) \mid \neg p \mid p \wedge p \mid p \vee p \mid \exists x \in set.p \mid \forall x \in set.p \mid set\ \Delta\ set \mid atomic \in set$
$-\Delta \equiv \subset | \subseteq | \not\subset | \cap | \cup$

message being communicated between two devices. S is a set of scenarios in the system. Each scenario, s, represents a set of actions happening via messaging among IoT devices in the smart home. $TeSA$ function defines a (set) of scenario(S) which would be triggered as the result of a given triggering event. As far as the triggering event is still effective, the triggered scenario(s) would also be going on and considered to be *active*. Priority of each scenario is the same as its trigger, and the priority value of messages included in one scenario would be the same as their containing scenario. Priorities are defined as binary relations between different triggers.

Check Access Predicate. The Check Access predicate includes three parts, each of which is responsible for a portion of access control in our extended model, and access would be granted to communicated the desired message if and only if all three following functions return TRUE:

1) $CheckAtt(s : D, m : M, r : D, current : ES)$ is a function checking the feasibility of communication based on

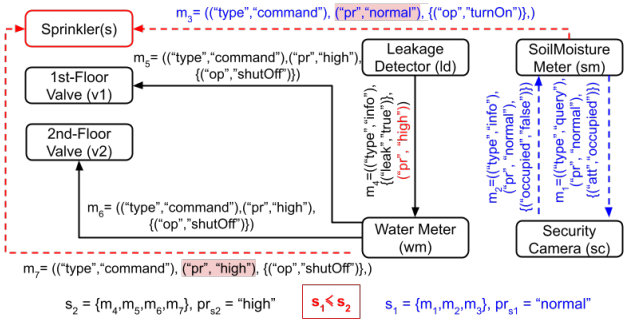

Fig. 4. Smart Home Use Case for Scenario-Driven D2D Communication

the message type and typeSetAtt in the message. It does the same checks as described in Section III-B.

2) $CheckPriority(s : D, m : M, r : D, current : ES)$ is specific to the scenario-based model proposed in this section. As messages may conflict only if they are of type *command*, this function returns TRUE, for all messages of type *info* or *query*. For command messages, it returns FALSE when the requested operation via command message is in conflict with the current operation being done at the moment on the receiver or if the message to be sent is assigned with lower priority. Otherwise, it returns TRUE.
3) $Authorization(s : D, m : M, r : D, current : ES)$ is a logical proposition which determines the set of allowed communications in the smart home IoT environment between different devices.

C. Smart Home Use Case

Figure 4 represents a smart home use case for our scenario-based model. *Sender* and *receiver* fields of messages are implied in the figure, arrows representing the direction of communications. This use case includes a two-story smart home IoT containing a soil moisture meter, an outdoor camera, a leakage detector, a sprinkler and two valves for first and second floor water flow control. A Soil moisture meter determines when the sprinkler starts spraying water by monitoring moisture level, and the leakage detector is responsible to cut off the water when any abnormal flow is discovered. We chose to exemplify scenario-based model by a different use case in which different devices are messaging each other. While in use case is Section III outdoor camera was responsible for all communications. Notably, none of the proposed models, not message-based, nor scenario based, need a central device to be responsible for all message communications.

Here, there are two scenarios going on. Scenario s_1 is initiated when the Soil Moisture Meter senses the soil moisture level to be lower than a specified threshold, so it sends a *query* message to the Security Camera in the backyard to inquire the outdoor's vacancy status, determined by *occupied* attribute value. If the backyard is vacant ($occupied = false$), then Soil Moisture Meter sends a *command* message with *normal* priority to *turn on* the Sprinkler. After a while when the sprinker is still spraying water, scenario s_2 is triggered when the Leakage Detector detects a leak, so it sends a *info* message to the Main Water Meter. Thereafter, Main Water Meter sends a *command* message with *high* priority to first/second-floor Valves, and Sprinkler to shut the flow off. Here the message to shut off the Sprinkler would go through because the $currenPR$ is *normal* and the message from the Main Water Meter has higher priority. So, Sprinkler faces a conflict as it has received two conflicting commands from Soil Main Moisture Meter and Leakage Detector. In our model, Sprinkler executes whatever operation included in the *command* message with higher priority, which is shut off from Leakage Detector. Our proposed access control could be configured as presented in Table IV to achieve following goals:

1) Authorize Soil Moisture Meter and Leakage detector to initiate appropriate set of actions by activating above-mentioned scenarios.
2) Enabling any receiver of conflicting *command* messages to resolve the conflict relying on priorities.

The smart home IoT devices, their attributes and available operations, and environment attribute and state are represented as *core components* of our model. The set of attributes for each device are represented under *attribute functions*. Five different priorities have been defined as a binary relation between triggers. There are two different scenarios, s_1 and s_2, each of which is a set of actions. The order of communicated messages in each scenario is indicated by their subscript number. Based on assigned priorities to triggers, which are $(LeakageDetector, (leak, true))$ with *high* priority and $(SoilMoistureMeter, droughtStatus = "dry"))$ with *normal* priority, indicated by prA function. The two scenarios of s_1 and s_2 have the same priorities as their triggering events and are comparable as $s_1 \prec s_2$. Check Access predicates are presented in Table IV, which include policy rules for CheckAtt, CheckPriority and Authorization of message communication between IoT device pairs.

V. Enforcement Architecture

Direct D2D communication is a long haul [12]. We present a decentralized enforcement architecture for our model, in which different IoT devices delegate the authorization decisions to an external entity. Even if heterogeneous IoT devices could directly intercommunicate, reliance upon an external entity to embrace decentralized access control is still desirable, as restricted IoT devices would no longer need to maintain/apply authorization information/rules themselves [71]. Our enforcement architecture is depicted in Fig. 5, which is designed based on AWS IoT because of its simplicity, security, flexibility, and being agnostic to device type and OS [9]. However, our enforcement architecture is not peculiar to AWS and could be designed independently of any cloud provider.

We deployed our architecture utilizing AWS Greengrass (GG) SDK [10], which lets users run a local event-driven computation unit (Lambda function [11]), messaging, data caching,

[9]https://aws.amazon.com/iot-device-management/
[10]https://docs.aws.amazon.com/greengrass/
[11]https://aws.amazon.com/lambda

TABLE IV
SCENARIO-BASED D2D MODEL: SMART HOME USE CASE

Core Components
D = {LeakageDetector, MainWaterMeter, SoilMoistureMeter, SecurityCamera, Sprinkler, FirstFloorValve, SecondFloorValve}
OP = {StartRecording, StopRecording, StartMeasure, StopMeasure, StartMonitor, StopMonitor, TurnOn, ShutOff}
ES = {current}
DOA = {(SecurityCamera,{StartRecording, StopRecording}), (LeakageDetector,{StartMonitor, StopMonitor}), (MainWaterMeter, {StartMeasure, StopMeasure}), (SoilMoistureMeter, {StartMonitor, StopMonitor}), (Sprinkler, {TurnOn, ShutOff}), (FirstFloorValve, {TurnOn, ShutOff}), (SecondFloorValve, {TurnOn, ShutOff})}
DAA = {(SecurityCamera,{id, type, location, recording, occupied}), (LeakageDetector,{id, type, leak}), (MainWaterMeter, {id, type, Measuring}), (SoilMoistureMeter, {id, type, droughtStatus}), (FirstFloorValve, {id, type, location, flowStatus}), (SecondFloorValve, {id, type, location, flowStatus}), (Sprinkler, {id, type, location, flowStatus})}
EAA = {day, time}
TE = {(LeakageDetector,(leak,"true")), (SoilMoistureMeter,(droughtStatus,"dry"))}
Ent = $D \cup ES \cup M \cup S \cup TE$

Attribute Functions
$(pr, \prec) = (\bot \prec low \prec normal \prec high \prec \top)$
id(SecurityCamera) = "sc", type(SecurityCamera) = "cameras", location(SecurityCamera) = "outdoor", recording(SecurityCamera) = {"true","false"}, occupied(SecurityCamera) = {"true","false"}, conflict(SecurityCamera)={(StartRecording,StopRecording)}
id(LeakageDetector) = "ld", type(LeakageDetector)= "detectors", subtype(LeakageDetector)="leak", leak(LeakageDetector) = {"true","false"}, conflict(LeakageDetector)={(StartMonitor, StopMonitor)}
id(MainWaterMeter) = "wm", type(MainWaterMeter)= "valves", subtype(MainWaterMeter)="watermeter",Measuring(MainWaterMeter) = {"true", "false"}, type(MainWaterMeter)= "valves", conflict(MainWaterMeter)={(StartMeasure, StopMeasure)}
id(SoilMoistureMeter) = "sm", type(SoilMoistureMeter)= "detectors", droughtStatus(SoilMoistureMeter) = {"dry","moist"}, conflict(SoilMostureMeter)={(StartMonitor, StopMonitor)}
id(FirstFloorValve) = "v1", type(FirstFloorValve)="valves", location(FirstFloorValve)="indoor", flowStatus(FirstFloorValve) = {"flowing", "halted"}, conflict(FirstFloorValve)={(TurnOn, ShutOff)}
id(SecondFloorValve) = "v2", conflict(SecondFloorValve)={(TurnOn, ShutOff)}, flowStatus(SecondFloorValve) = {"flowing", "halted"},
id(SecondFloorValve) = "v1", type(SecondFloorValve)="valves", location(SecondFloorValve)="indoor", flowStatus(SecondFloorValve) = {"flowing", "halted"}, conflict(SecondFloorValve)={(TurnOn, ShutOff)}
id(Sprinkler) = "s", type(Sprinkler)="valves", location(Sprinkler)="outdoor", flowStatus(Sprinkler) = {"flowing", "halted"}, conflict(Sprinkler)={(TurnOn, ShutOff)}
$-attAssignType(d, att) = \begin{cases} static & \text{if } att \in \{"id", "type", "subtype", "location"\} \\ dynamic & \text{otherwise} \end{cases}$

Message, Scenarios and Auxiliary Functions
$-(pr, \prec) = (\bot \prec low \prec normal \prec high \prec \top)$
$-prA = \{((LeakageDetector, leak = "true"), high),$
$((SoilMoistureMeter, droughtStatus = "dry"), normal)\}$
$-A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$
$-a_1 = (SoilMoistureMeter, \{("type", "query"), ("att", "occupied")\}, SecurityCamera)$
$-a_2 = (SecurityCamera, \{("type", "info"), ("occupied", "false")\}, SoilMoistureMeter)$
$-a_3 = (SoilMoistureMeter, \{("type", "command"), ("op", "TurnOn")\}, SecurityCamera)$
$-a_4 = (LeakageDetector, \{("type", "info"), ("leak", "true")\}, MainWaterMeter)$
$-a_5 = (MainWaterMeter, \{("type", "command"), ("op", "ShutOff")\}, FirstFloorValve)$
$-a_6 = (MainWaterMeter, \{("type", "command"), ("op", "ShutOff")\}, SecondFloorValve)$
$-a_7 = (MainWaterMeter, \{("type", "command"), ("op", "ShutOff")\}, Sprinkler)$
$-S = \{s_1, s_2\}$
$-s_1 = \{Action_{s_1}, pr_{s_1}, tr_{s_1}, active, id\}$, $Action_{s_1} = \{a_1, a_2, a_3\}, tr_{s_1} = (SoilMoistureMeter, (droughtStatus, "dry"), pr_{s_1} = "normal"$
$-s_2 = \{Action_{s_2}, pr_{s_2}, tr_{s_2}, active, id\}$, $Action_{s_2} = \{a_4, a_5, a_6, a_7\}, tr_{s_2} = (LeakageDetector, (leak, "true"), pr_{s_2} = "high"$
$-id(s_1) = s, id(s_2) = s'$
$-active(s_1)$ and $active(s_2)$ is determined based on definition in Table III, at each instant of time.
$-M = \{m_1, m_2, m_3, m_4, m_5, m_6, m_7\}$
$m_1 = (("type", "query"), ("pr", msgPrA(sender_1, m_1, receiver_1)), \{("att", "occupied")"\})$
$m_2 = (("type", "info"), ("pr", msgPrA(sender_2, m_1, receiver_2)), \{("occupied", "false")\})\}$
$m_3 = (("type", "command"), ("pr", msgPrA(sender_3, m_3, receiver_3)), \{("op", "turnOn")\})\}$
$m_4 = (("type", "info"), ("pr", msgPrA(sender_4, m_4, receiver_4)), \{("leak", "true")"\})$
$m_5 = (("type", "command"), ("pr", msgPrA(sender_5, m_5, receiver_5)), \{("op", "shutOff")"\})$
$m_6 = (("type", "command"), ("pr", msgPrA(sender_6, m_6, receiver_6)), \{("op", "shutOff")"\})$
$m_7 = (("type", "command"), ("pr", msgPrA(sender_7, m_7, receiver_7)), \{("op", "shutOff")"\})$
-After s_1's activation: $\begin{cases} msgPrA(m_1) = msgPrA(m_2) = \bot \\ msgPrA(m_3) = "normal" \end{cases}$
-After s_2's activation: $\begin{cases} msgPrA(m_4) = \bot \\ msgPrA(m_5) = msgPrA(m_6) = msgPrA(m_7) = "high" \end{cases}$

TABLE IV
SCENARIO-BASED D2D MODEL: SMART HOME USE CASE (CONT.)

Attribute Authorization Function

$-CheckAccess(s:D, m:M, r:D, current:ES) \equiv$
$CheckAtt(s:D, m:M, r:D, current:ES) \wedge$
$CheckPriority(s:D, m:M, r:D, current:ES) \wedge$
$Authorization(s:D, m:M, r:D, current:ES)$

$$-CheckAtt(s:D, m:M, r:D, current:ES) = True \iff typeSetAtt(m) = \begin{cases} \subseteq 2^{DAA(r)} & \text{if } m.value_1 = \text{"}query\text{"} \\ \in DOA(r) & \text{if } m.value_1 = \text{"}command\text{"} \\ \subseteq 2^{DAA(s)} & \text{if } m.value_1 = \text{"}info\text{"} \end{cases}$$

$-CheckPriority(SoilMoistureMeter, m_1, \text{"}sc\text{"}, \{current\}) \equiv \text{"}true\text{"}$
$-CheckPriority(SecurityCamera, m_2, \text{"}sm\text{"}, \{current\}) \equiv \text{"}true\text{"}$
$-CheckPriority(SoilMoistureMeter, m_3, \text{"}s\text{"}, \{current\}) \equiv \text{"}true\text{"}$
$-CheckPriority(LeakageDetector, m_4, \text{"}wm\text{"}, \{current\}) \equiv \text{"}true\text{"}$
$-CheckPriority(MainWaterMeter, m_5, \text{"}v1\text{"}, \{current\}) \equiv \text{"}true\text{"}$
$-CheckPriority(MainWaterMeter, m_6, \text{"}v2\text{"}, \{current\}) \equiv \text{"}true\text{"}$
$-CheckPriority(MainWaterMeter, m_7, \text{"}s, \{current\}) \equiv \text{"}true\text{"}$

$-Authorization(s:D, m:M, r:D, current:ES) \equiv$
$q_1 \vee q_2 \vee q_3 \vee q_4 \vee q_5$
$q_1 = \left[\left(m.att_1 = \text{"}info\text{"}\right) \wedge \left(typeSetAtt(m) \in \{\text{"}leak\text{"}\}\right) \wedge \left(type(r) = \text{"}valves\text{"}\right) \wedge \left(subtype(r) = \text{"}watermeter\text{"}\right) \wedge \left(type(s) = \text{"}detectors\text{"}\right)\right]$
$q_2 = \left[\left(typeSetAtt(m) \in \{\text{"}ShutOff, TurnOn\text{"}\}\right) \wedge \left(m.att_1 = \text{"}command\text{"}\right) \wedge \left(type(r) = type(s) = \text{"}valves\text{"}\right) \wedge \left(\text{subtype(s)="watermeter"}\right)\right]$
$q_3 = \left[\left(m.att_1 = \text{"}query\text{"}\right) \wedge \left(typeSetAtt(m) \in \{\text{"}occupied\text{"}\}\right) \wedge \left(type(s) = \text{"}detectors\text{"}\right) \wedge \left(subtype(s) = \text{"}soil\text{"}\right) \wedge \left(type(r) = \text{"}cameras\text{"}\right) \wedge \left(location(r) = \text{"}outdoor\text{"}\right)\right]$
$q_4 = \left[\left(m.att_1 = \text{"}info\text{"}\right) \wedge \left(typeSetAtt(m) \in \{\text{"}occupied\text{"}\}\right) \wedge \left(type(s) = \text{"}camera\text{"}\right) \wedge \left(location(source) = \text{"}outdoor\text{"}\right) \wedge \left(type(r) = \text{"}detectors\text{"}\right) \wedge \left(subtype(r) = \text{"}soil\text{"}\right)\right]$
$q_5 = \left[\left(typeSetAtt(m) \in \{\text{"}TurnOn, ShutOff\text{"}\}\right) \wedge \left(m.att_1 = \text{"}command\text{"}\right) \wedge \left(type(s) = \text{"}detectors\text{"}\right) \wedge \left(type(r) = \text{"}valves\text{"}\right) \wedge \left(subtype(s) = \text{"}soil\text{"}\right) \wedge \left(location(r) = \text{"}outdoor\text{"}\right)\right]$

and synchronization for connected devices in a secure way without the need for computation on cloud providers. GG communicates with locally connected devices over MQTT [12] protocol which is a lightweight machine-to-machine publish-subscribe protocol designed for constrained devices. Using MQTT, devices will communicate on their private topics, *device/shadow/update*, to update their shadows, and trigger the lambda function. As devices communicate their state on their private topics, it would be stored locally, as the device shadow, which is accessible only by themselves and the lambda function. When lambda code is triggered, it executes the code we developed to check the policy. If permission granted, the results will be communicated, the shadow states will be updated, and the physical device will respond accordingly.

[12]http://www.ibm.com/developerworks/webservices/library/ws-mqtt/index.html

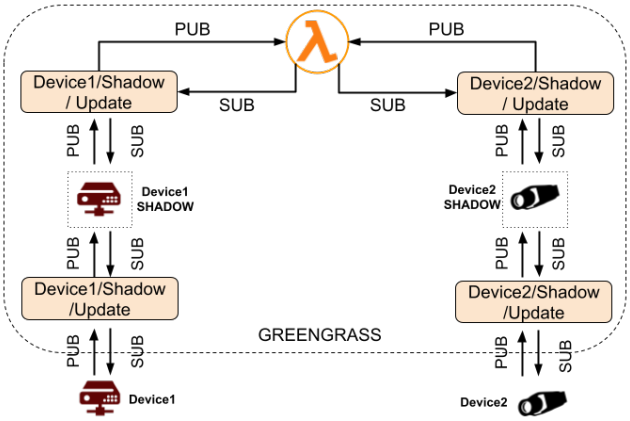

Fig. 5. Device-to-Device Architecture

TABLE V
FULL EXPERIMENT AND DEVICE STATE UPDATE STATISTICS.

Expr.	Min	10%	25%	Med	75%	90%	95%	99%	Max
ft	24.2	25.4	26.0	34.9	36.7	39.0	42.9	47.4	67.2
su	3.5	3.6	3.7	5.7	6.6	6.9	7.4	10.2	21.8

VI. IMPLEMENTATION

A. Experiment Setup

A proof-of-concept implementation of two smart home use cases in Sec. III-C and IV-C are described. We use AWS IoT GG V1, running on a dedicated virtual machine with one virtual CPU, 2 GB of RAM and 20 GB hard drive. The operating system of the virtual machine is Ubuntu 20.4.2 LTS and it is connected to a 1 Gbps network. We wrote the AWS lambda code on GG in Python 3.8, which is running in a long-lived isolated docker environment with 64 MB RAM. GG lambda is running locally on the same network as the devices and not in the cloud.

B. Implementation Results

Both scenarios we implemented have a similar computational process. A device sends a message to another device through the GG lambda function. Upon receiving the message, lambda spins up multiple threads, and begins processing our code. This initial process results in a few spurious results that have particularly long processing delays shown at the top 1% of results depicted in Table V. As a message received by lambda, once the sender and destination devices are authenticated, the code would proceed. In general, lambda inquires the shadow status of the device(s) as necessary, the priority levels of messages, then sends a series of commands to various devices based on the *policy.json*. Any priority conflicts between *command* messages received by the same device are resolved based on this rule: the higher priority is dominant. If two commands with the same priorities are conflicting, we run the most recent one. If the scenario did not exist in the policy, the situation is considered invalid, disregarded, and no additional commands would be executed.

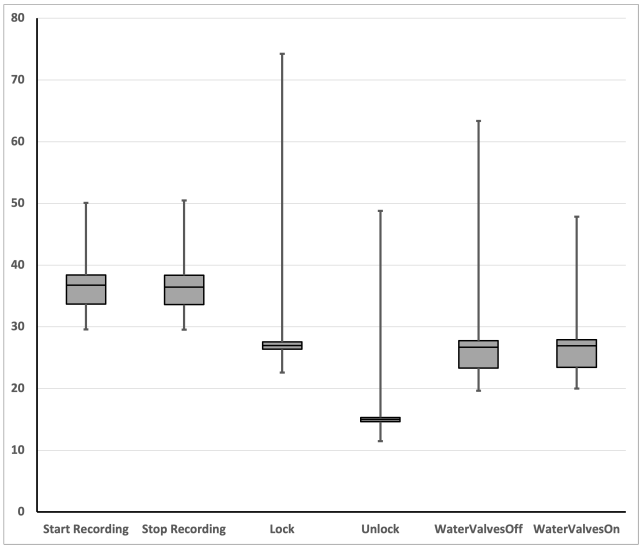

Fig. 6. Time (ms) per Action on GG

Our implementation results show an average time for a device to execute an action, have lambda process it, and update the respective devices to be on average 35 milliseconds. This complete process is indicated as 'full timer' (ft), averaged over 500 trials and includes transmission time going to/from lambda to the devices over a 1 Gbps network connection. A more detailed breakdown of the analysis is included in Table V. The average time to update a device's state (su), e.g., a door lock going from unlocked to locked, is 6ms. The various use cases and actions are also timed and shown as boxplots in Fig. 6, where the whiskers are the minimum and maximum values collected across 500 trials for each experiment. The box shows the 25 percentile and 75 percentile of the data, with median as the line contained inside the box.

VII. Discussion and Research Directions

A. Proposed Model/Architecture Properties

Our model is unparalleled as it provides specification for mediating access in device-to-device communications for the first time. Besides being context-aware, dynamic and lightweight, following features are notable about our model.

Continuity of Access. Attributes of sender/receiver devices may change at any time as a result of an operation or an alteration in the environment, a.k.a mutability [72]. We utilized AWS for our model enforcement, in which lambda would be informed on any changes in participating device attributes via subscription to corresponding topics. Any change triggers the lambda which then re-evaluates the policy and adjusts the access authorizations accordingly. So, the continuity is supported using lambda in AWS architecture. In order to provide continuity as one of the model's elements, regardless of its enforcement method, the continuous retrieval and evaluation of entity/environment attributes should be incorporated in the model. Moreover, continuity of access control is concluded when the proposed model is able to revoke the previously granted access in case of any unintended change in attributes.

Architecture Agnostic. AWS as the enforcement architecture augmented our framework with some desired features, however, our model is not peculiar to it.

B. Research Agenda

A factual interoperability solution for IoT environments would be obtained when there is no need to rely on a gateway for communication of heterogeneous IoT devices. Heterogeneity in different layers of IoT communications, data/information models, communication syntax/semantics/technologies, etc. should also be investigated [28], [57]. There are many IoT devices which are called black box, as they provide least visibility toward their internal state and composition. So, they cannot be managed and accessed as traditional IT devices [1]. Building access control solutions which are tailored to specific features of IoT devices and environments is an utmost requirement. Proposed access control models are better to comprise of both user-to-device and device-to-device communications in order to be considered a thorough solution.

VIII. Conclusion

In this paper we proposed a novel device-to-device access control model specification based on attribute-based access control (ABAC) approach. Our model relies on the message passing paradigm for authorization of intercommunication among IoT devices in the smart home environment. We also introduced the concept of scenarios, reflecting a chain of actions initiated by a triggering event. A total order relation among triggersto address the probable conflicts is another contribution of our paper. We backed up our model by proposing an enforcement architecture which is appropriate for intermittent connections in the smart home IoT by bringing the computations to the edge. Our proof-of-concept implementation endorses our proposed model to be efficient and appropriate.

Acknowledgement

This work is partially supported by NSF CREST Grant 1736209.

References

[1] K. Megas *et al.*, "Establishing confidence in iot device security: How do we get there?" NIST, Tech. Rep., 2021.
[2] W. Zhou *et al.*, "Discovering and understanding the security hazards in the interactions between IoT devices, mobile apps, and clouds on smart home platforms," in *USENIX*, 2019.
[3] Z. Ling *et al.*, "Security vulnerabilities of internet of things: A case study of the smart plug system," *IoT*, 2017.
[4] S. Notra *et al.*, "An experimental study of security and privacy risks with emerging household appliances," in *CNS*. IEEE, 2014.
[5] A. L. M. Neto *et al.*, "Aot: Authentication and access control for the entire IoT device life-cycle," in *SenSys*. ACM, 2016.
[6] B. Fouladi *et al.*, "Honey, i'm home!!, hacking zwave home automation systems," *Black Hat USA*, 2013.
[7] R. Goyal *et al.*, "Mind the tracker you wear: a security analysis of wearable health trackers," in *SAC*. ACM, 2016.
[8] E. Ronen *et al.*, "IoT goes nuclear: Creating a ZigBee chain reaction," in *SP*. IEEE, 2017.
[9] Z. B. Celik *et al.*, "Sensitive information tracking in commodity IoT," in *USENIX*, 2018.

[10] S. Lee *et al.*, "FACT: Functionality-centric access control system for IoT programming frameworks," in *SACMAT*. ACM, 2017.
[11] E. Fernandes *et al.*, "Security analysis of emerging smart home applications," in *SP*. IEEE, 2016.
[12] H. Tschofenig *et al.*, "Architectural considerations in smart object networking," *RFC 7452*, 2015.
[13] M. Noura *et al.*, "Interoperability in internet of things: Taxonomies and open challenges," *Mobile networks and applications*, 2019.
[14] O. Bello *et al.*, "Intelligent device-to-device communication in the internet of things," *Systems*, 2014.
[15] IEEE, "Standard for an architectural framework for the internet of things (IoT)." [Online]. Available: https://standards.ieee.org/ieee/2413/6226/
[16] C. E. R. Results, "symbIoTe-symbiosis of smart objects across IoT environments." [Online]. Available: https://cordis.europa.eu/project/id/688156
[17] ——, "Building an iot open innovation ecosystem for connected smart objects." [Online]. Available: https://cordis.europa.eu/project/id/688203
[18] S. Soursos *et al.*, "Towards the cross-domain interoperability of IoT platforms," in *EuCNC*. IEEE, 2016.
[19] S. Behrad *et al.*, "A new scalable authentication and access control mechanism for 5G-based IoT," *Future Generation Computer Systems*, 2020.
[20] B. a. Seok, "Secure D2D communication for 5G IoT network based on lightweight cryptography," *Applied Sciences*, 2020.
[21] D. Bonino *et al.*, "The DOG gateway: enabling ontology-based intelligent domotic environments," *Transactions on Consumer Electronics*, 2008.
[22] V. Peláez *et al.*, "Multilevel and hybrid architecture for device abstraction and context information management in smart home environments," in *AmI*. Springer, 2010.
[23] J. E. Kim *et al.*, "Seamless integration of heterogeneous devices and access control in smart homes and its evaluation," *Intelligent Buildings International*, 2017.
[24] EPI, "Open virtual neighbourhood network to connect IoT infrastructures and smart objects." [Online]. Available: https://vicinity2020.eu/
[25] Y. Guan *et al.*, "An open virtual neighbourhood network to connect IoT infrastructures and smart objects—Vicinity: IoT enables interoperability as a service," in *GIoTS*. IEEE, 2017.
[26] A. Cimmino *et al.*, "VICINITY: IoT semantic interoperability based on the web of things," in *DCOSS*. IEEE, 2019.
[27] J. Manyika *et al.*, "Unlocking the potential of the internet of things," *McKinsey Global Institute*, 2015.
[28] G. Fortino *et al.*, "Towards multi-layer interoperability of heterogeneous IoT platforms: The INTER-IoT approach," *Integration, interconnection, and interoperability of IoT systems*, 2018.
[29] K. Shafique *et al.*, "Internet of things for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT scenarios," *Access*, 2020.
[30] A. Ouaddah *et al.*, "Access control in the internet of things: Big challenges and new opportunities," *Networks*, 2017.
[31] S. Ravidas *et al.*, "Access control in internet-of-things: A survey," *Network and Computer Applications*, 2019.
[32] E. Bertin *et al.*, "Access control in the internet of things: A survey of existing approaches and open research questions," *Annals of telecommunications*, 2019.
[33] B. Cremonezi *et al.*, "Survey on identity and access management for internet of things," 2020.
[34] J. Qiu *et al.*, "A survey on access control in the age of internet of things," *IoT*, 2020.
[35] Z. N. Mohammad *et al.*, "Access control and authorization in smart homes: A survey," *Tsinghua Science and Technology*, 2021.
[36] B. Ur *et al.*, "The current state of access control for smart devices in homes," in *HUPS*, 2013.
[37] W. He *et al.*, "Rethinking access control and authentication for the home internet of things (IoT)," in *USENIX*, 2018.
[38] E. Zeng *et al.*, "Understanding and improving security and privacy in multi-user smart homes: a design exploration and in-home user study," in *USENIX*, 2019.
[39] M. Tabassum *et al.*, "Smart home beyond the home: A case for community-based access control," in *CHI*. ACM, 2020.
[40] S. Kanchi *et al.*, "A multi perspective access control in a smart home," in *CODASPY*. ACM, 2021.
[41] S. Ameer *et al.*, "The egrbac model for smart home IoT," in *IRI*. IEEE, 2020.
[42] A. K. Sikder *et al.*, "Kratos: Multi-user multi-device-aware access control system for the smart home," in *WiSec*. ACM, 2020.
[43] M. Alramadhan *et al.*, "An overview of access control mechanisms for internet of things," in *ICCCN*. IEEE, 2017.
[44] F. Bakir *et al.*, "Caplets: Resource aware capability-based access control for IoT," in *SEC*. IEEE, 2021.
[45] S. Dutta *et al.*, "Context sensitive access control in smart home environments," in *BigDataSecurity, HPSC and IDS*. IEEE, 2020.
[46] S. Ameer *et al.*, "The HABAC model for smart home iot and comparison to EGRBAC," in *Sat-CPS*. ACM, 2021.
[47] ——, "An attribute-based approach toward a secured smart-home iot access control and a comparison with a role-based approach," *Information*, 2022.
[48] A. Dorri *et al.*, "Blockchain for iot security and privacy: The case study of a smart home," in *PerCom Workshops*. IEEE, 2017.
[49] B. Mbarek *et al.*, "Blockchain-based access control for IoT in smart home systems," in *DEXA*. Springer, 2020.
[50] A. Qashlan *et al.*, "Security and privacy implementation in smart home: Attributes based access control and smart contracts," in *TrustCom*. IEEE, 2020.
[51] V. Hu, "Blockchain for access control systems," NIST, Tech. Rep., 2021.
[52] B. Tang *et al.*, "Iot passport: A blockchain-based trust framework for collaborative internet-of-things," in *SACMAT*. ACM, 2019.
[53] B. Bezawada *et al.*, "Securing home iot environments with attribute-based access control," in *ABAC*. ACM, 2018.
[54] J. Fan *et al.*, "Ruledger: Ensuring execution integrity in trigger-action iot platforms," in *INFOCOM*. IEEE, 2021.
[55] S.-R. Oh *et al.*, "An interoperable access control framework for diverse iot platforms based on OAuth and role," *Sensors*, 2019.
[56] B. Anggorojati *et al.*, "Capability-based access control delegation model on the federated IoT network," in *WPMC*. IEEE, 2012.
[57] C. Perera *et al.*, "Context-aware dynamic discovery and configuration of 'things' in smart environments," in *Big Data and Internet of Things: A Roadmap for Smart Environments*. Springer, 2014.
[58] O. Bello *et al.*, "Intelligent device-to-device communication in the internet of things," *Systems*, 2014.
[59] L. Cruz-Piris *et al.*, "Access control mechanism for IoT environments based on modelling communication procedures as resources," *Sensors*, 2018.
[60] A. Alshehri *et al.*, "Access control models for cloud-enabled IoT: A proposed architecture and research agenda," in *CIC*. IEEE, 2016.
[61] S. Bhatt *et al.*, "ABAC-CC: Attribute-based access control and communication control for internet of things," in *SACMAT*. ACM, 2020.
[62] G. D. Abowd *et al.*, "Towards a better understanding of context and context-awareness," in *HUC*. Springer.
[63] G. Zhang *et al.*, "An extended role based access control model for the internet of things," in *ICINA*. IEEE, 2010.
[64] P. Colombo *et al.*, "Access control enforcement within MQTT-based internet of things ecosystems," in *SACMAT*. ACM, 2018.
[65] K. Guesmia *et al.*, "Orbac from access control model to access usage model," *Applied Intelligence*, 2018.
[66] Y. Dong *et al.*, "Contexts-states-aware access control for internet of things," in *CSCWD*. IEEE, 2018.
[67] D. Meyer *et al.*, "A threat-model for building and home automation," in *INDIN*. IEEE, 2016.
[68] D. Geneiatakis *et al.*, "Security and privacy issues for an iot based smart home," in *MIPRO*. IEEE, 2017.
[69] D. Dolev *et al.*, "On the security of public key protocols," *Transactions on information theory*, 1983.
[70] G. S. Poh *et al.*, "Privhome: Privacy-preserving authenticated communication in smart home environment," *TDSC*, 2019.
[71] V. Beltran *et al.*, "Overview of device access control in the IoT and its challenges," *Communications Magazine*, 2018.
[72] R. Sandhu *et al.*, "Usage control: A vision for next generation access control," in *MMM-ACNS*. Springer, 2003.