Dynamic Groups and Attribute-Based Access Control for Next-Generation Smart Cars

Maanak Gupta, James Benson, Farhan Patwa and Ravi Sandhu

Institute for Cyber Security (ICS),
Center for Security and Privacy Enhanced Cloud Computing (C-SPECC),
Department of Computer Science, University of Texas at San Antonio, San Antonio, Texas, USA
Email: gmaanakg@yahoo.com, {james.benson, farhan.patwa, ravi.sandhu}@utsa.edu

ABSTRACT

Smart cars are among the essential components and major drivers of future cities and connected world. The interaction among connected entities in this vehicular internet of things (IoT) domain, which also involves smart traffic infrastructure, restaurant beacons, emergency vehicles, etc., offer several real-time applications and provide safer and pleasant driving experience to consumers. With more than 100 million lines of code and hundreds of sensors, these connected vehicles (CVs) expose a large attack surface, which can be remotely compromised and exploited by malicious attackers. Security and privacy are big concerns that deter the adoption of smart cars, which if not properly addressed will have grave implications with risk to human life and limb. In this paper, we present a formalized dynamic groups and attribute-based access control (ABAC) model (referred as CV-ABAC$_G$) for smart cars ecosystem, where the model not only considers system wide attributes-based security policies but also takes into account the individual user privacy preferences for allowing or denying service notifications, alerts and operations to on-board resources. Further, we introduce a novel notion of groups in vehicular IoT, which are dynamically assigned to moving entities like connected cars, based on their current GPS coordinates, speed or other attributes, to ensure relevance of location and time sensitive notification services, to provide administrative benefits to manage large numbers of entities, and to enable attributes inheritance for fine-grained authorization policies. We present proof of concept implementation of our model in AWS cloud platform demonstrating real-world uses cases along with performance metrics.

CCS CONCEPTS

• **Security and privacy** → **Formal security models**; **Access control**; **Authorization**; *Security requirements*; *Domain-specific security and privacy architectures*;

KEYWORDS

Access Control, Smart Cars, Connected Vehicles, Internet of Things, Authorization, Attribute-Based Access Control, Amazon Web Services (AWS), Autonomous Cars, Security, Privacy, Cloud Computing

1 INTRODUCTION

Internet of Things (IoT) has become a dominant technology which has proliferated to different application domains including healthcare, homes, industry, power-grid, to make lives smarter. It is predicted [2] that the global IoT market will grow to $457 Billion by year 2020, attaining a compound annual growth rate of 28.5%. Automation is leading the world today, and with 'things' around sensing and acting on their own or with a remote user command, has given humans to have anything accessible with a finger touch. Data generated by these smart devices unleash countless business opportunities and offer customer targeted services. IoT along with 'infinite' capabilities of cloud computing are ideally matched with desirable synergy in current technology-oriented world, which has been often termed as cloud-enabled, cloud-centric or cloud-assisted IoT in literature [12, 17, 18, 38].

IoT is embraced by every industry with automobile manufacturers and transportation among the most aggressive. Vehicular IoT inherits intrinsic IoT characteristics but dynamic pairing, mobility of vehicles, real-time, location sensitivity are some features which separates it from common IoT applications. The vision of smart city incorporates intelligent transportation where connected vehicles can 'talk' to each other (V2V) and exchange information to ensure driver safety and offer location-based services. These intelligent vehicles can also interact with smart roadside infrastructure (V2I), with pedestrian on road (V2H) or send data to the cloud for processing. Basic safety messages (BSMs) are exchanged among entities using commonly used WiFi like secure and reliable Dedicated Short Range Communication (DSRC) protocol. Vehicles can receive speed limit notification and flash flood alerts on car dashboard or via seat vibration. A car will receive information about nearby parking garages, restaurant offers or remote engine monitoring by authorized mechanic with nearby repair facility and discounts updating automatically. These services will provide pleasant travel experience to drivers and unleash business potential in this intelligent transportation domain. Smart internet connected vehicles embed softwares having more than 100 million lines of code to control critical systems and functionality, with plethora of sensors and electronic control units (ECUs) on board generating huge amounts of data so these vehicles are often termed as 'datacenter on wheels'.

As vehicles get exposed to external environment and internet, they become vulnerable to cyber attacks. Common security vulnerabilities including buffer overflow, malware, privilege escalation, and trojans etc. can be exploited in connected vehicles. Other potential threats include untrustworthy or fake messages from smart objects, malicious software injection, data privacy, ECU hacking and control, and spoofing connected vehicle sensor. With broad attack surface exposed via air-bag ECU, On-Board Diagnostics (OBD) port, USB, Bluetooth, remote key, and tire-pressure monitoring system etc. these attacks have become much easier to orchestrate. In-vehicle Controller Area Network (CAN) bus also needs security to protect message exchange among ECUs. Further, communication with external networks including cellular, WiFi and insecure public networks of gas stations, toll roads, service garages, or after-market dongles are a big threat to connected vehicles security. Cyber incidents including Jeep [54] and Tesla Model X [51] hacks where engine was stopped and steering remotely controlled demonstrate security vulnerabilities. Smart car incidents have serious implications as they can even result in loss of human life.

Access control [22, 46, 47] mechanisms are widely used to restrict unauthorized access to resources and secure communication among entities. Attribute-based access control (ABAC) [35, 37] provides finer granularity and offers flexibility in distributed multi-entity communication scenarios, which considers characteristics of participating entities along with system and environment properties to determine access decision. Smart cars ecosystem involves dynamic interaction and message exchange among connected objects, which must be authorized. It is necessary that only legitimate entities are allowed to control on-board sensors, data messages and send notifications. Further, user-centric privacy requires that users can control what alerts they want to receive, what advertisements they are interested or who can access their car's sensors, etc. This paper focuses on the access control needs in connected smart cars and proposes an attribute-based access control model for connected vehicles[1] ecosystem, referred as CV-ABAC$_G$. Our solution considers the attributes of moving entities like current location, speed etc. to dynamically assign them to various groups (predefined by smart city administration), for implementing attributes-based security policies, and also incorporate user-specific privacy preferences for ensuring relevance of notifications service in constantly changing and mobile smart cars ecosystem. We implemented a prototype of our model as an external authorization engine hooked into the widely used AWS (Amazon Web Services) cloud platform [3].

Rest of the paper is organized as follows. Section 2 discusses related work and reviews the extended access control architecture (E-ACO) recently proposed for vehicular IoT environment. Section 3 highlights authorization requirements and emphasize the need of dynamic groups in smart cars applications. Section 4 presents and formalizes our proposed groups and attribute-based access control model (CV-ABAC$_G$) for connected vehicles ecosystem. Section 5 provides AWS implementation of dynamic groups assignment of entities based on attributes and discusses our external policy decision and enforcement engine along with performance metrics. Section 6 summarizes our work.

---

[1]In this paper, we use the terms smart cars and connected vehicles interchangeably which also subsumes autonomous vehicles.
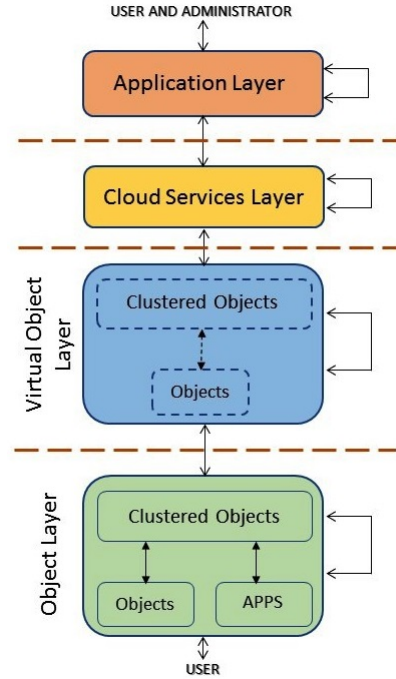

Figure 1: E-ACO Architecture [31]

2 RELATED WORK

Vehicular IoT and smart cars involve dynamic communications and data exchange which requires access controls to restrict within authorized entities. In this section, we first discuss a recently proposed extended access control architecture (E-ACO) which focuses on access control requirements in connected vehicles. We also highlight some important work done by government and private agencies to gauge cyber risks and security measures in smart vehicles.

2.1 Extended ACO Architecture

Several IoT architectures with multi-layer stack have been discussed in literature [13–15, 19, 25]. Alsehri and Sandhu [14] recently presented a general IoT architecture which includes virtual objects [42] and cloud as two middleware layers. Virtual objects resolve IoT issues of heterogeneity and connectivity whereas on-demand capabilities of cloud are in cloud service layer. Gupta and Sandhu [31, 32] extended this IoT architecture for specific vehicular IoT and connected vehicles domain. This extended access control architecture (E-ACO), shown in Figure 1, introduced clustered objects (like smart cars and traffic lights) which are objects with multiple individual sensors. Also, these clustered objects have applications (for example, lane departure or safety warning system in cars) installed on board, which is usually not the case in general IoT realm.

As shown in Figure 1, four layered E-ACO has **Object Layer** at the bottom which represents physical clustered objects and sensors along with applications installed on them. In-vehicle communication at this layer is mainly supported by Ethernet and CAN technologies, whereas communication across clustered objects is done using DSRC (used for BSM exchange in V2V communication), WiFi,

or LTE etc. It should be noted that each layer in E-ACO architecture interacts within itself and with entities in adjacent layers. Therefore, object layer will interact with users at the bottom and virtual object layer above it. The **Virtual Object Layer** acts as an intermediate between cloud services and physical layer, which offers the necessary abstraction by creating cyber entities for physical objects in object layer. In particular in connected vehicles domain, where cars are moving across different terrains where internet connectivity can be an issue, it is important to have cyber entities which maintain the state of the corresponding physical object as best known and to be updated when connectivity is restored. When two sensors s_1 and s_2 across different vehicles interact with each other, the order of communication using virtual objects will follow s_1 to vs_1 (virtual entity of s_1), vs_1 to vs_2 and vs_2 to physical sensor s_2. **Cloud Services and Application Layer:** As applications use cloud services, therefore these two layers are discussed together. On-board sensors generate data which is stored and processed by cloud services, which is used by applications to offer services to end-users. Cyber entities of physical objects can be created in cloud layer which provides a persistent state information of objects. It is important to mention that central cloud may incur latency and bandwidth issues in time-sensitive applications which can be resolved by introducing edge or fog computing infrastructure.

2.2 Relevant Background

Smart cars and associated applications are still in early stages but involve some established technologies. Vehicular Ad-hoc Networks (VANETs) [11] have been discussed which support vehicle to vehicle and infrastructure communication for user services. In VANETS, moving cars and infrastructure act as network nodes to provide storage, computation and other services. This concept is further extended with the inclusion of cloud computing. Vehicular Clouds (VC) [20, 24, 43] were proposed to integrate VANETs and cloud, to offer on-the fly edge/cloud platform to cars and applications by utilizing on-board resources. VCs are relevant in smart cars real-time and location-centric applications and services, which are otherwise impractical due to latency and bandwidth issues of central cloud. Several VC architectures have been discussed including stationary, fixed infrastructure or dynamic [36, 53].

Cyber threats to connected vehicles are very serious concerns. Government agencies and private sectors are well aware of the risks involved and want to ensure that no open doors are left to orchestrate attacks before wide adoption. The US Department of Transportation (USDOT) has invested in Intelligent Transportation System (ITS) [16] which has connected vehicles as an important component with aim to reduce accidental fatalities. Cyber security is a key area and along with National Highway Traffic Safety Administration (NHTSA), it has released cyber-security guidelines [40, 41]. Security Credential Management System (SCMS) [52] is proposed as DSRC message security solution in vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication. It uses Public Key Infrastructure (PKI)-based approach to enable trusted interaction where a certificate authority issued certificate is attached to each BSM [1] to ensure vehicle trustworthiness. US Government Accountability Office (GAO) [23] have widely discussed vulnerabilities and attack surfaces in smart vehicles, and also proposed
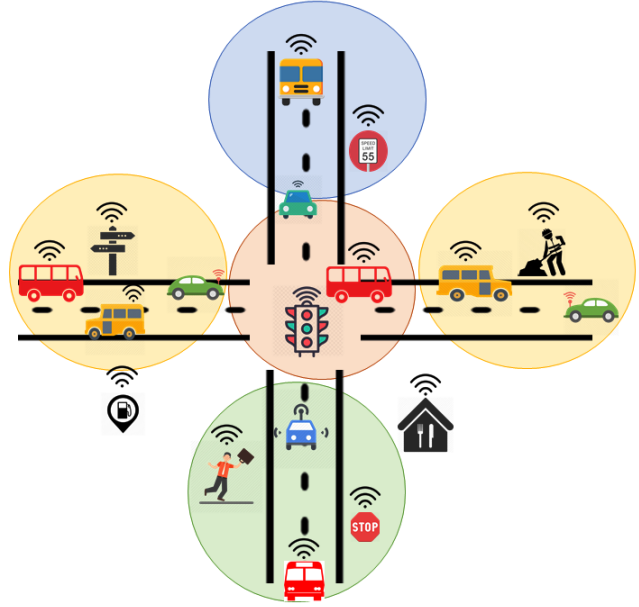

Figure 2: Smart City with Location Groups

solutions to prevent such threats. European Union Agency for Network and Information Security (ENISA) also studied critical assets and threats in smart cars together with security mechanisms to mitigate them [21]. Cooperative Intelligent Transport Systems (C-ITS) for European Union [49, 50] has defined a PKI-based trust model to ensure authenticity and integrity of vehicle messages.

Homomorphic encryption based security solutions and protocols have been extensively discussed to provide location proximity [33, 39, 55] which can help to provide location based services without sharing the exact coordinates of drivers. These approaches can be used and complement our proposed CV-ABAC$_G$ model to resolve the privacy concerns of end users.

Access controls are widely used in computer systems to restrict unauthorized access to resources. Park et al [44, 45] proposed an activity centric access control model for social networks which considers user privacy policies in access decision. CV-ABAC$_G$ model is inspired from this work besides being a pure ABAC model with dynamic groups which are pertinent in smart cars ecosystem.

3 AUTHORIZATION REQUIREMENTS IN SMART CARS

Smart cars expose the conventionally isolated car systems to external environment via internet. The dynamic and short-lived real time V2V and V2I interaction with entities in and around connected vehicle needs to ensure message confidentiality and integrity, as also protection of on-board resources from adversaries. This section provides an overview of access control requirements and underlines the need for dynamic groups in smart vehicles IoT domain.

3.1 Multi-Layer and User Privacy Preferences

Broad attack surface of connected vehicles is the first entry point to in-vehicle critical systems. We believe that two level access control

policies are the minimum essential to protect the external interface and internal ECU communication. Access control for external environment will protect on-board sensors, applications and user personal data from unauthorized access by entities including vehicles, applications, masquerading remote mechanics or other adversaries. Over-the air firmware update needs to be checked and must be allowed only from authorized sources. An attacker even if successful in passing through the first check point, must be restricted at the in-vehicle level, which secures overwrite and control of critical units (engine, brakes, telematics etc.) from adversaries. Vehicles exchange BSMs which raises an important question about trust. It must be ensured that information received is correct and from a trusted party, before being used by on-vehicle applications. Applications access sensors within and outside the car, which must be authorized, for example, a lane departure warning system accessing tire sensors must be checked to prevent a spoofed application reading vehicle movements. A passenger accessing infotainment (information and entertainment) systems of the car via Bluetooth or using smartphone inside car must also be authorized.

Smart cars location-based services enable notifications and alerts to vehicles. A user must be allowed to set his personal preferences whether he wants to receive advertisements or filter out which ones are acceptable. For instance, a user may not want to receive restaurant notifications but is interested in flash-flood warnings. System wide policy, like a speed warning to all over-speeding vehicles or a policy of who can control speed of autonomous car are needed.

Data protection in cloud is critical due to frequent occurrence of data breaches. Big Data access control [26–29] is essential when user privacy has to be ensured and unauthorized disclosure is not allowed. Cross cloud trust models are needed to allow data access when mechanic application in private cloud reads data in car-manufacturer cloud. Physical tampering of vehicle OBD and sensors also require protection but is out of scope for this paper.

3.2 Relevance of Groups

Most smart cars applications and service requests from drivers are location specific and time sensitive. For example, a driver might want to get warning signals when traveling near a blind spot, in school zone or pedestrians crossing road. Further, notifications sent to drivers are short-lived and mostly pertinent around current GPS coordinates. A gas discount notification from a nearby station, an accident warning two blocks away or ice on the bridge, are some example where alerts are sent to all vehicles in the area. Therefore, we believe that dynamically categorizing connected vehicles into location groups will be helpful for scoping the vehicles to be notified instead of a general broadcast and reduce administrative overheads, since single notification for the group will trigger alerts for all its members. Also, entities present at a location have certain characteristics (like stop sign warning, speed limit, deer-threat etc.) in common, which can be inherited by being a group member. Figure 2 represents how various smart entities can be separated into different location groups defined by appropriate authorities in a smart city system. These groups are dynamically assigned to connected vehicles based on their attributes, personal preferences, interests or current GPS coordinates as further elaborated in the model and implementation section discussed later.

Figure 3: Representative Groups Hierarchy

Groups hierarchy can also exist, as shown in Figure 3, with sub-groups within a larger parent group so as to reduce the number of vehicles to be notified. For instance, under location group, sub-groups can be created for cars, buses, police vehicles or ambulances, to enable targeted alerts to ambulances or police vehicle sub-groups defined within the location group. Groups can be defined based on services, for example, a group of cars within the car parent group which take part in car-pooling (CP) service or those which want to receive gas station offers. Group hierarchy [30, 48] also enables attributes inheritance from parent to child groups.

4 ACCESS CONTROL MODEL FOR CONNECTED VEHICLES ECOSYSTEM

Dynamic communication and data exchange among entities in connected vehicles ecosystem require multi-layer access control policies, which are managed centrally and also driven by individual user preferences. Therefore, an access control model must incorporate all such user and system requirements and offer fine-grained authorization solutions. In this section, we will discuss and formally define our proposed connected vehicle attribute-based access control model with dynamic groups, which we refer as CV-ABAC$_G$.

4.1 CV-ABAC$_G$ Model Overview

The conceptual CV-ABAC$_G$ model is shown in Figure 4 with formal definitions summarized in Table 1. The basic model has following components: Sources (S), Clustered Objects (CO), Objects in clustered objects (O), Groups (G), Operations (OP), Activities (A), Authorization Policies (POL), and Attributes (ATT).

Sources (S): These entities initiate activities (explained below) on various smart objects, groups and applications in the ecosystem. A source can be a user, an application, administrator, sensor, hand-held device, clustered object (such as a connected car), or a group defined in the system. For instance, in case of flash flood or deer threat warning, activity source is police or city department triggering an alert to all vehicles in the area. Similarly, car mechanic is a
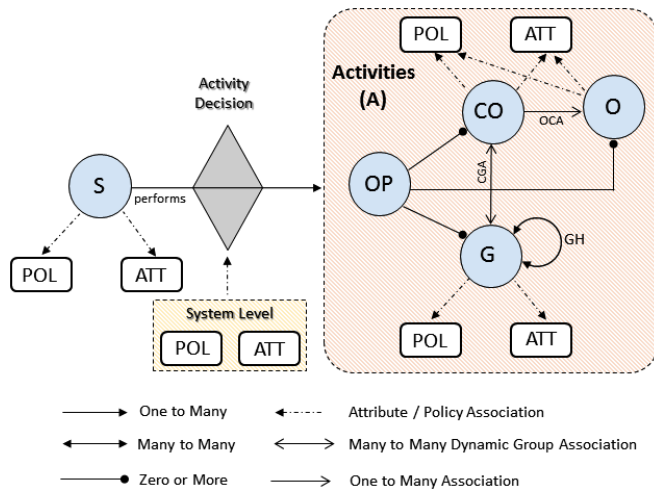
Figure 4: A Conceptual CV-ABAC$_G$ Model

source, when he tries to access data from on-board engine sensor in the car using his remote cloud based application. A restaurant or gas-station issuing coupons are also considered as source.

Clustered Objects (CO): Clustered objects are particularly relevant in case of connected vehicles, traffic lights or smart devices held by humans as they have multiple sensors and actuators. A smart car with on-board sensors, ECUs (like tire pressure, lane departure, or engine control) and applications is a clustered object. These smart entities interact and exchange data among themselves and with others such as requestor source, applications or cloud. An important reason to incorporate clustered objects is to reflect cross-vehicle and intra-vehicle communication. The fact that two smart vehicles can exchange basic safety messages (BSM) with each other shows clustered object communication.

Objects in clustered objects (O): These are individual sensors, ECUs and applications installed in clustered objects. Objects in smart cars include sensors for internal state of the vehicle, e.g., engine diagnostics, emission control, cabin monitoring system, as well as sensors for external environment such as cameras, temperature, rain, etc. Control commands can directly be issued to these objects, and data can be read remotely. Applications (like lane departure warning system) on board can also access data from these objects to provide alerts to driver or to a remote service provider.

Groups (G): A group is a logical collection of clustered objects with similar characteristics or requirements. With these groups, subset of COs can be sent relevant notification and also attributes can be assigned to group members. Some groups which can be defined smart vehicles ecosystem include location specific groups, service specific groups (like car-pooling, gas station promotions etc.) or vehicle type (a group of cars, buses etc.). Group hierarchy (GH) also exists which enables attributes and policies inheritance from parent to children groups. For simplicity, we require that a vehicle or CO can be direct member of only one group at same hierarchy level. For example, a car can be in either location A or B group and but not both. Such restriction helps in managing attributes inheritance and enhances the usability of our model.

Operations (OP): These are actions which can be performed against clustered objects, individual objects or groups. Examples include: a mechanic performing read, write or control operations on engine ECU, a restaurant triggering notifications to vehicles in location A group. Operations also include administrative actions like creating or updating attributes or policies for COs, objects and groups, which are usually performed by system/security administrators.

Activities (A): Activities encompass both operational and administrative activities which are performed by various sources in the system. An activity can have one or many atomic operations (OP) involved and will need authorization policies, which can be user privacy preferences, system defined or both, to allow or deny an activity. For example, a car pooling notification activity generated by a requestor (source) will be broadcast to all relevant vehicles in the locations nearby using location groups, however individual drivers must also receive or respond to that request based on individual preferences. A driver may not want to car-pool the requestor because of poor rating or because he is not going to the destination the requestor asked for. Therefore, an activity can involve multiple set of policies defined at different levels which must be evaluated, in car-pooling case a policy is set to determine cars to be notified and then driver personal preferences. We have primarily divided these smart car activities into following categories.

- Service Requests: These are activities initiated by entities or users (via applications). For instance, a vehicle break-down initiates a service request to other vehicles around, or a user using a smartphone initiates a car-pooling requests for a destination to cars which are available for the service.
- Administration: These activities perform administrative operations in system which include changing policies and attributes of entities or determining the group hierarchy. It also defines the scope of groups, how user privacy preferences are used, or how vehicles are determined to be a member of a group etc.
- Notifications: These are group centric activities where all members are notified for any updates about the group (like speed limit or deer threat notifications in location A) or for locations-based marketing promotions by parking lots or restaurants.
- Control and Usage: These activities include simple read, write or control operations performed remotely or within a vehicle. Over the air updates issued by manufacturer or turning on car climate control using a smart key are remote activities whereas a passenger accessing infotainment system using smartphone and on-board car applications reading car camera are local.

Authorization Policies and Attributes: CV-ABAC$_G$ model incorporates individual user privacy controls for different entities by managing authorization policies and entity attributes. A shown in Figure 4 policy of sources include personal preferences, whereas attributes reflect characteristics like name, age or gender. Policies can be defined for clustered objects, for instance, a USB can be plugged only by car owner, or which mechanic can access an on-board sensor. Attributes of a car include GPS coordinates, speed, heading direction, and vehicle size. Groups also set policies and attributes for themselves, for example, car pooling group policy of who can be its member. Similarly, system wide policies are also considered, for instance, policy to determine which groups will be sent information when a request comes from a source, or policy to

Table 1: Formal CV-ABAC$_G$ Model Definitions for Connected Vehicles Ecosystem

---

Basic Sets and Functions

– S, CO, O, G, OP are finite sets of sources, clustered objects, objects, groups and operations respectively [blue circles in Figure 4].

– A is a finite set of activities which can be performed in system.

– ATT is a finite set of attributes associated with S, CO, O, G and system-wide.

– For each attribute att in ATT, Range(att) is a finite set of atomic values.

– attType: ATT = {set, atomic}, defines attributes to be set or atomic valued.

– Each attribute att in ATT maps entities in S, CO, O, G to attribute values. Formally,

$$\text{att} : S \cup CO \cup O \cup G \cup \{\text{system-wide}\} \to \begin{cases} \text{Range(att)} \cup \{\bot\} & \text{if attType(att) = atomic} \\ 2^{\text{Range(att)}} & \text{if attType(att) = set} \end{cases}$$

– POL is a finite set of authorization policies associated with individual S, CO, O, G.

– directG : CO → G, mapping each clustered object to a system group, equivalently CGA ⊆ CO × G.

– parentCO : O → CO, mapping each object to a clustered object, equivalently OCA ⊆ O × CO.

– GH ⊆ G × G, a partial order relation $\geq_g$ on G. Equivalently, parentG : G → 2^G, mapping group to a set of parent groups in hierarchy.

Effective Attributes of Groups, Clustered Objects and Objects (Derived Functions)

– For each attribute att in ATT such that attType(att) = set:

  - $\text{effG}_{\text{att}} : G \to 2^{\text{Range(att)}}$, defined as $\text{effG}_{\text{att}}(g_i) = \text{att}(g_i) \cup (\bigcup_{g \in \{g_j | g_i \geq_g g_j\}} \text{effG}_{\text{att}}(g))$.

  - $\text{effCO}_{\text{att}} : CO \to 2^{\text{Range(att)}}$, defined as $\text{effCO}_{\text{att}}(co) = \text{att}(co) \cup \text{effG}_{\text{att}}(\text{directG}(co))$.

  - $\text{effO}_{\text{att}} : O \to 2^{\text{Range(att)}}$, defined as $\text{effO}_{\text{att}}(o) = \text{att}(o) \cup \text{effCO}_{\text{att}}(\text{parentCO}(o))$.

– For each attribute att in ATT such that attType(att) = atomic:

  - $\text{effG}_{\text{att}} : G \to \text{Range(att)} \cup \{\bot\}$, defined as $\text{effG}_{\text{att}}(g_i) = \begin{cases} \text{att}(g_i) & \text{if } \forall g' \in \text{parentG}(g_i). \text{ effG}_{\text{att}}(g') = \bot \\ \text{effG}_{\text{att}}(g') & \text{if } \exists \text{ parentG}(g_i). \text{ effG}_{\text{att}}(\text{parentG}(g_i)) \neq \bot \text{ then select} \\ & \text{parent } g' \text{ with effG}_{\text{att}}(g') \neq \bot \text{ updated most recently.} \end{cases}$

  - $\text{effCO}_{\text{att}} : CO \to \text{Range(att)} \cup \{\bot\}$, defined as $\text{effCO}_{\text{att}}(co) = \begin{cases} \text{att}(co) & \text{if effG}_{\text{att}}(\text{directG}(co)) = \bot \\ \text{effG}_{\text{att}}(\text{directG}(co)) & \text{otherwise} \end{cases}$

  - $\text{effO}_{\text{att}} : O \to \text{Range(att)} \cup \{\bot\}$, defined as $\text{effO}_{\text{att}}(o) = \begin{cases} \text{att}(o) & \text{if effCO}_{\text{att}}(\text{parentCO}(o)) = \bot \\ \text{effCO}_{\text{att}}(\text{parentCO}(o)) & \text{otherwise} \end{cases}$

Authorization Functions (Policies)

– Authorization Function: For each op ∈ OP, $\text{Auth}_{\text{op}}(s : S, ob : CO \cup O \cup G)$ is a propositional logic formula returning true or false, which is defined using the following policy language:

  - $\alpha ::= \alpha \wedge \alpha \mid \alpha \vee \alpha \mid (\alpha) \mid \neg\alpha \mid \exists\, x \in \text{set}.\alpha \mid \forall\, x \in \text{set}.\alpha \mid \text{set} \triangle \text{set} \mid \text{atomic} \in \text{set} \mid \text{atomic} \notin \text{set}$
  - $\triangle ::= \subset \mid \subseteq \mid \nsubseteq \mid \cap \mid \cup$
  - set $::= \text{eff}_{\text{att}}(i) \mid \text{att}(i)$ for att ∈ ATT, i ∈ S ∪ CO ∪ O ∪ G ∪ {system-wide}, attType(att) = set
  - atomic $::= \text{eff}_{\text{att}}(i) \mid \text{att}(i) \mid \text{value}$ for att ∈ ATT, i ∈ S ∪ CO ∪ O ∪ G ∪ {system-wide}, attType(att) = atomic

Authorization Decision

– A source s ∈ S is allowed to perform an activity a ∈ A, stated as Authorization(a : A, s : S), if the required policies needed to allow the activity are included and evaluated to make final decision. These multi-layer policies must be evaluated for individual operations (op$_i$ ∈ OP) to be performed by source s ∈ S on relevant objects (x$_i$ ∈ CO ∪ O ∪ G).

Formally, $\text{Authorization}(a : A, s : S) \Rightarrow \text{Auth}_{\text{op}_1}(s : S, x_1), \text{Auth}_{\text{op}_2}(s : S, x_2), \ldots\ldots\ldots\ldots, \text{Auth}_{\text{op}_n}(s : S, x_3)$

---

change group hierarchy. Policies also include attributes of entities involved in an activity. A CO can inherit attributes from dynamically assigned groups which will change as the CO leaves old group and adds to new group.

It should be noted that attributes of entities change more often than system wide or individual policies. Attributes are more dynamic in nature which are added or removed with the movement of vehicles or change in surroundings, like GPS coordinates or temperature. Policies once set by administrators or users are more static

and only the attributes which comprise the policy change the outcome of a policy but the policy definition remains relatively fixed. For instance, a user policy could state that 'Send restaurant notifications only from Cheesecake factory'. In such case, only attribute name of the restaurant sending the notification will be checked and if it is equal to Cheesecake factory will be able to advertise to that user. Dynamic policies are also possible, for instance, a policy may state that police vans in locations groups A and B are notified in case of emergency, but, in case of a bigger threat this policy can be changed or overwritten with police vans in groups A, B C and

D. Our model assumes that no policies or attributes are changed during an activity evaluation.

Some activities will need multi-level policy evaluation and may include user privacy preferences. For instance, a user must be allowed to decide if he wants to share data from car sensors or whether wants to get marketing advertisements. Each activity will evaluate required system and user policies to make final decision.

4.2 Formal Definitions

As shown in Table 1, sources, clustered objects, objects and groups can be directly assigned values from the set of atomic values (denoted by Range(att)) for attribute att in set ATT. Each attribute can be a set or atomic valued, determined by attType function and based on its type, entities can be assigned a single value including null ($\bot$) for an atomic attribute, or multiple values for set-valued attribute from the attribute range. POL is the set of authorization policies defined in the system which will be defined below.

Clustered objects can be members of different groups, based on preferences and requirements. For example, a car is assigned to a location group based on its GPS coordinates. In our model, we assume that a clustered object can be directly assigned to only one group at same hierarchy level (specified by directG function). As we will discuss later that since groups inherit attributes from parent groups, assigning a clustered object to one parent group is sufficient to realize attributes inheritance. Smart cars have sensors and applications installed in them, which can also be accessed by different sources. Therefore, parentCO function determines the clustered object to which an object belongs, which is a one to many mapping i.e an object can only belong to one CO while a CO can have multiple objects. Further, group hierarchy GH (shown as self loop on G), is defined using a partial order relation on G and denoted by $\succeq_g$, where $g_1 \succeq_g g_2$ signifies g_1 is child group of g_2 and g_1 inherits all the attributes of g_2. Function parentG computes the set of parent groups in hierarchy for a child group.

The benefit to introduce groups is ease of administration where multiple attributes can be assigned or removed from member clustered objects with single administrative operation. Group hierarchy enables attributes inheritance from parent to child groups. Therefore, in case of set valued attributes, the effective attribute att of a group g_i (denoted by $\text{effG}_{att}(g_i)$) is the union of directly assigned values for attribute att and the effective values for att for all its parent groups in group hierarchy. This definition is well formed since $\succeq_g$ is a partial order. For a maximal group g_j in this ordering, we have $\text{effG}_{att}(g_j) = att(g_j)$, giving base cases for this recursive definition. The effective attribute values of clustered object for attribute att (stated as effCO_{att}) will then be the directly assigned values for att and the effective attribute values of att for the group to which CO is directly assigned (by directG). Similarly, in addition to direct attributes, sensors in car can inherit attributes from the car itself (eg. make, model, location), effO_{att} calculates these effective attributes of objects. For set valued attributes, union operation will be sufficient which is not true for atomic attributes. In case of groups, the most recently updated non-null attribute values in parent groups will overwrite the values of child group as defined in Table 1. For example, if the most recent value updated in one of the parent groups for Deer_Threat attribute is 'ON', this value will
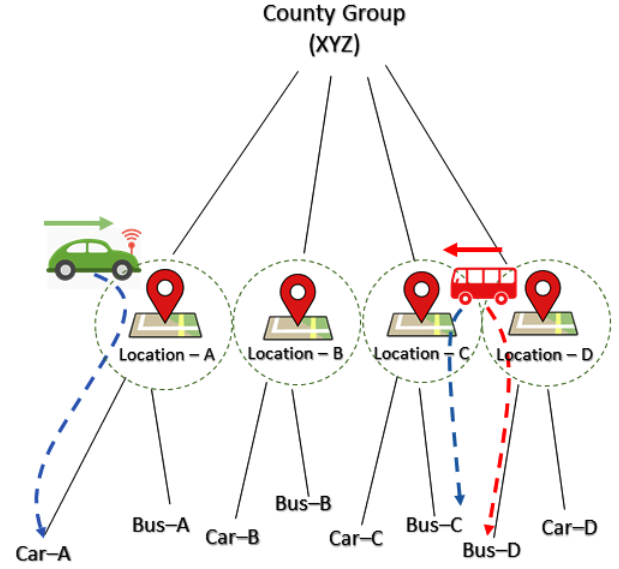

Figure 5: Groups Hierarchy in AWS

trickle to the child group. It should be noted that overwriting with the most recently updated value in groups is one of the many approaches to inherit atomic attributes, but for the dynamic nature of smart cars ecosystem, we believe this is most appropriate. Clustered object inherits non-null atomic value from its direct parent group as stated by $\text{effCO}_{att}(co) = \text{effG}_{att}(\text{directG}(co))$. In case of objects, parent clustered object will overwrite non-null atomic attributes. For atomic attributes, if the parent(s) has null value for an attribute, the entity (group, clustered object or object) will retain its directly assigned value without any overwrite.

Authorization functions are defined for each operation op $\in$ OP, which are policies defined in the system. POL is the set of all authorization functions, $\text{Auth}_{op}(s : S, ob : CO \cup O \cup G)$, which specify the conditions under which source s $\in$ S can execute operation op $\in$ OP on object ob $\in$ CO $\cup$ O $\cup$ G. Such policies include privacy preferences set by users for individual clustered object, objects and groups or can be system wide by security administrators. The conditions can be specified as propositional logic formula using policy language defined in Table 1. Multiple policies must be satisfied before an activity is allowed to perform. Authorization function, Authorization(a : A, s : S), where an activity a $\in$ A is allowed by source s $\in$ S, specifies the system level, user privacy policies or other relevant policies returning true for an activity to succeed.

CV-ABAC_G is an attribute-based access control model which satisfies fine-grained authorization needs of dynamic, location oriented and time sensitive services and applications in cloud assisted smart cars ecosystem. The model supports personalized privacy controls by utilizing individual user policies and attributes, along with dynamic groups assignment. Our model assumes that the information and attributes shared by source and object entities are trusted, for instance, location coordinates sent by a car are correct, and uses this shared information to make access and notification decisions. How to ensure that the information is from a trusted source or is correct is out of the scope of this work.
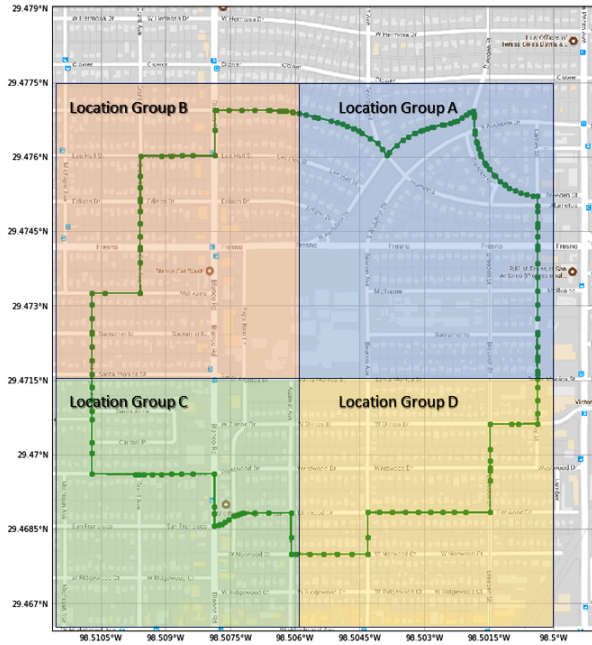
Figure 6: Vehicle GPS Coordinates and Groups Demarcation

5 CV-ABAC$_G$ ENFORCEMENT IN AWS

In this section, we present a proof of concept demonstration of CV-ABAC$_G$ model by enforcing a use case of smart cars using AWS IoT service [5]. The implementation will demonstrate how dynamic groups assignment and multi-layer authorization policies required in connected vehicle ecosystem can be realized in AWS. We have used simulations to reflect real connected smart vehicles, however, it does not undermine the plausibility, use and advantage of our proposed model as further elaborated in following discussion. It should be noted that no long term vehicle data including real-time GPS coordinates are collected in central cloud, which mitigates user privacy concerns and encourages wide adoption of the model.

5.1 Description of Use Cases

Location based alerts and notifications are important in smart cars applications and motivate our use cases. We will build upon our defined group hierarchy in AWS shown in Figure 5. Our implementation will enforce access controls and service notification relevance in following use cases:

Deer Threat Notification - Smart infrastructure in the city can sense the surrounding environment and notify group(s) regarding the change. In this use case, a motion sensor senses deers in the area and changes Deer_Threat attribute of location group to ON which in-turn sends alerts to all member vehicles in that location. Similar, implementation can be done in case of accident notification, speed limit warning or location based marketing.

Car-Pooling - A traveller needs a ride to Location-A. Using a mobile application, he sends car-pooling requests to vehicles in his vicinity which are heading to the destination location asked by the traveller. The request is received by AWS cloud, which computes location and appropriate groups based on the coordinates of the

```
('Received new coordinates from:', 'Vehicle-1')
Sun May 27 02:56:30 2018
  Location A
    Car-A : [u'Vehicle-1', u'Vehicle-2']
    Bus-A : []
  Location B
    Car-B : []
    Bus-B : [u'Vehicle-6']
  Location C
    Car-C : [u'Vehicle-3', u'Vehicle-4']
    Bus-C : []
  Location D
    Car-D : []
    Bus-D : [u'Vehicle-5']
```

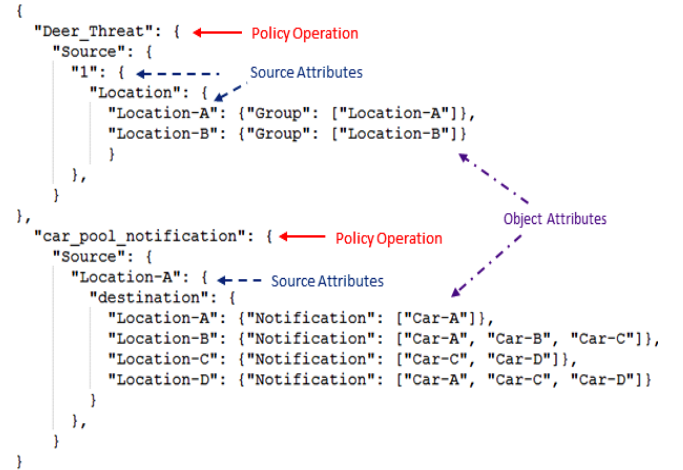**Figure 7: Dynamic Groups and Vehicles in AWS**


Figure 8: Attribute Based Policies in AWS

requester, to publish notifications to nearby cars. All the members of group Car-A, B, C or D can get the request, but some cars may not want to be part of car-pooling, or do not want some requestors to join them because of ratings. User policies will be also checked before a driver is notified of likely car-pool customer.

5.2 Prototype Implementation

AWS implementation of our model in these use-cases involves two phases: administrative phase and operational phase. Administrative part involves creation of groups hierarchy, dynamic assignment of moving vehicles to different location and sub-groups, attributes inheritance from parent to child groups and to group members, and attributes modification of entities. Operational part covers how groups are used to scope down the number of vehicles who receive messages or notifications from different sources. Both these phases involve multi-layer access control polices. We created an ABAC policy decision (PDP) and enforcement point (PEP) [34], and implemented our external policy evaluation engine which is hooked with AWS to enable attribute-based authorization.

Administrative Phase: We created a group hierarchy in AWS as shown in Figure 5. In this hierarchy, County-XYZ is divided into four disjoint Location-A, B, C and D groups, with each having Car and Bus subgroups for vehicle type car or bus. We created 10
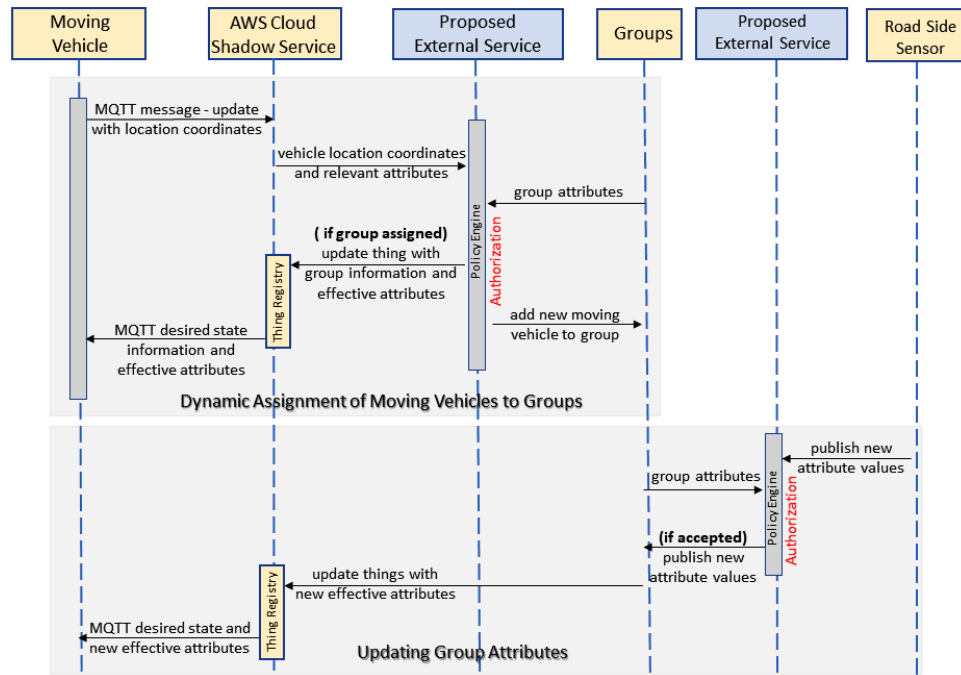
Figure 9: Sequence Diagram for Dynamic Groups and Attributes Assignment in AWS

vehicles and simulated their movement using a python script which publishes MQTT message to shadows of these vehicles with current GPS coordinates (generated using Google API [10]) iterated over green dots shown in Figure 6. The area was demarcated into four locations and a moving vehicle belongs to a subgroup in one of these groups. Assuming current location of Vehicle-1 as Location-D, and it publishes MQTT message with payload:

```
{"state": {"reported": {"Latitude": "29.4769353",
                        "Longitude":"-98.5018237"}}}
```

to AWS topic: $aws/things/Vehicle-1/shadow/update, its new location changes to Location-A and since we defined the vehicle type as car, it is assigned to Car-A group under Location-A as shown by snippet in Figure 7. Both attributes, vehicle type and current coordinates of vehicle, are used to dynamically assign groups, which is important in moving smart vehicles. These functionalities are implemented as a stand alone service (can be enforced as a Lambda service [6] function) using Boto [7] which is the AWS SDK for Python. Further, in case of deer threat notification use-case, we simulated a location-sensor which senses deers in the area and updates the attribute 'Deer_Threat' of location group to 'ON' or 'OFF', which is then notified to all members of location and its subgroups. We defined an attribute-based policy to control which sensors can change the 'Deer_Threat' attribute of location groups. As shown in Figure 8, our policy for Deer_Threat operation checks that a motion sensor with ID = '1' and current groups of Location-A can update the attribute Deer_Threat for group Location-A, and if sensor is relocated to Location-B it can update attribute for Location-B group only. This policy ensures that the sensor must be in that location group for which it is updating Deer_Threat attribute.

The complete sequence of events performed in AWS along with our stand-alone service for the administrative phase is shown in Figure 9. A moving vehicle updates its coordinates to AWS shadow service, which along with attributes of vehicles and location groups determines if the vehicle can be member of the group using our external enforcement service. If authorization policy allows vehicle to be a member of group, the vehicle and group is notified and vehicle inherits all attributes of its newly assigned group. Similarly, if attribute 'Deer_Threat' of group is allowed (by authorization policy) to be changed by the location sensor, the new values are propagated to all its members. We implemented attribute inheritance from parent to child groups through our service using `update_thing_group` and `update_thing` methods. In our use-case attributes inheritance exist from Location-A to all both subgroups Car-A and Bus-A, and to vehicles in Car-A and Bus-A. Therefore, when attribute 'Deer_Threat' is set to ON in group Location-A, its new attributes using Boto `describe_thing_group` command are:

```
{'Center-Latitude': '29.4745', 'Center-Longitude':
            '-98.503','Deer_Threat': 'ON'}
```

This inherits the attributes to Car-A child group whose effective attributes will now be:

```
{'Center-Latitude': '29.4745', 'Center-Longitude':
'-98.503','Deer_Threat': 'ON', 'Location': 'A'}
```

As shown in Figure 7, both Vehicle-1 and Vehicle-2 as member of Car-A, the effective attributes of Vehicle-2 are:

```
{'Center-Latitude': '29.4745', 'Center-Longitude':
'-98.503','Deer_Threat': 'ON', 'Location': 'A',
'Type': 'Car', 'VIN': '9246572903752', 'thingName':
'Vehicle-2'}
```
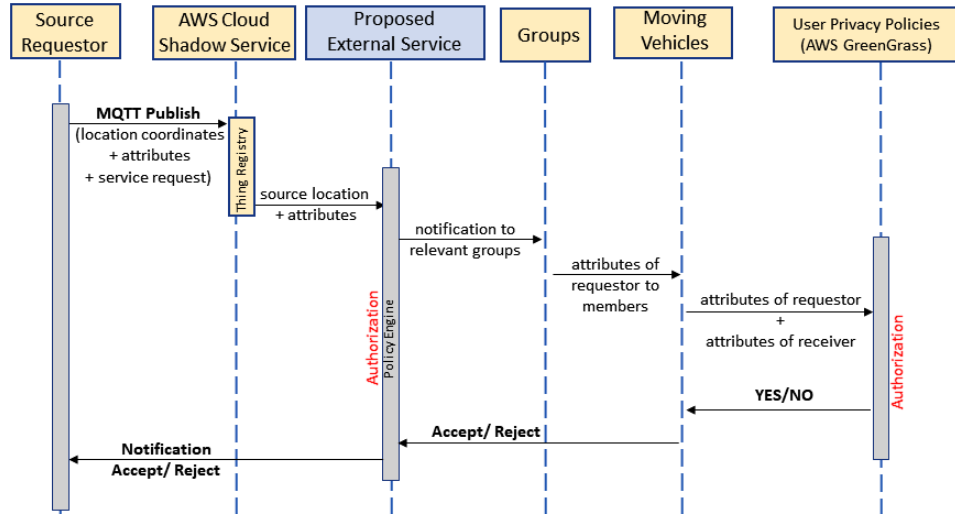
Figure 10: Sequence Diagram for Attributes Based Authorization in AWS

Operational Phase: In this phase, attribute-based policies are used to restrict service and notification activities which may require single or multi-level policies along with user preferences. In car-pooling use case, we defined policies to restrict notifications to only a subset of relevant vehicles in specific locations. We simulated requestor in AWS needing car-pool. It has attribute 'destination' with value in Location-A, B, C or D. Requestor sends current and destination location as MQTT message to AWS topic $aws/things/Requestor/shadow/update which based on these attributes determine subgroups to which service requests is sent.

```
{"state": {"reported": {"policy":
 "car_pool_notification", "source": "Location-A",
 "destination": "Location-B"}}}
```

The policy for `car_pool_notification` operation (shown in Figure 8) suggests that if current location of source requestor is 'Location-A' and destination location is somewhere in 'Location-A' then all members of sub-group Car-A should be notified. Similarly, if the destination attribute is Location-B, then all members of Car-A, Car-B and Car-C needs notification. In our use-case, all members of these sub-groups are notified. The policy restricts the number of vehicles which will be requested as compared to all vehicles getting irrelevant notification (as they are far from the requestor or are not vehicle type car) and illustrates the importance of location-centric smart car ecosystem. Similarly, location-based marketing can be restricted and policies can be defined to control such notifications.

User privacy policies take into effect once the subset of vehicles is calculated. These policies encapsulate user preferences, for instance, in car pooling a particular driver is not going to the destination requested by the requestor in his request or a driver do not want restaurant advertisements, therefore such notifications will not be displayed on his car dashboard. These local policies are implemented using AWS Greengrass [4] which allows to run local lambda functions on the device (in our case a connected vehicle) to enable edge computing facility, an important requirement in real-time smart car applications and enforce privacy policies. Once

Number of Requests	Policy Enforcer Execution Time (in ms)
10	0.0501
20	0.1011
30	0.1264
40	0.1630
50	0.1999

| nth Request	Cars Notified	
	With ABAC Policy	Without Policy
41st	2	5
42nd	3	5
43rd	5	5
44th	3	5
45th	2	5
46th	3	5

Figure 11: Policy Enforcement Time and Scoping

accepted by drivers, a SNS (AWS Simple Notification Service) [8] message can be triggered for requestor from accepting vehicles along with name and vehicle number. The sequence of events for car-pooling activity and multi-layer authorization policies together with user personal preferences is shown in Figure 10.

Our proposed external service to implement ABAC policy decision and evaluation helps achieve fine grained authorization needed in smart cars ecosystem. The implementation also demonstrates dynamic groups assignment based on mobile vehicle GPS coordinates and attributes along with groups based attributes inheritance which offer administrative benefits in enforcing an ABAC model. In this entire implementation, no persistent data from moving vehicles is collected or stored by the central authority hosted cloud which reaffirms its privacy preserving benefits. Note that the use-cases discussed to enforce CV-ABAC$_G$ are not real-time and can bear some latency due to the use of cloud infrastructure. Although our CV-ABAC$_G$ enforcement in AWS reflects its use for cloud based applications, we believe similar model can also be implemented in edge (or fog) systems as well to cater more real-time use-cases.

5.3 Performance Evaluation

We evaluated the performance of our proposed CV-ABAC$_G$ model in AWS and provide different metrics when no policies were used
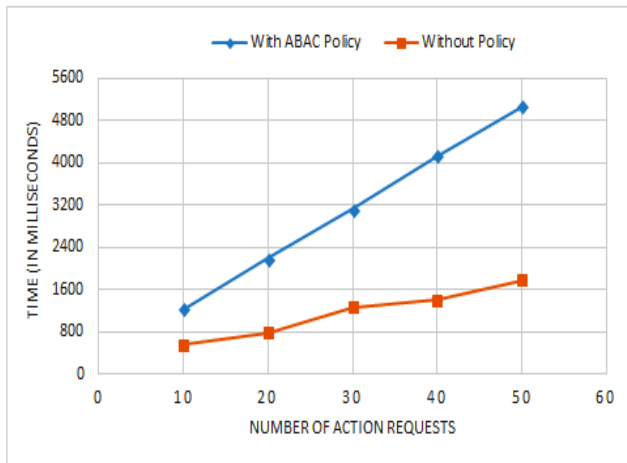
Figure 12: Performance Evaluation

against our implemented ABAC policies for the car-pooling notification use-case. As shown in Figure 11, our external policy evaluation engine has average time (in milliseconds) to decide on car-pooling service requests and provide the subset of cars which are notified. This scoping ensures the service relevance as without a policy all 5 vehicles were sent car-pool request (even when one was 20 miles away from the requestor), whereas with attribute based policies only nearby cars are notified. The performance graph shown in Figure 12 compares no policy execution time (red line) against implemented ABAC policy (blue line). Since, in our experiments the policy (shown in Figure 8) evaluated for each access requests is the same, we get a linear graph as the number of access requests increase the number of times the policy is evaluated and so its total evaluation time. Some variation in red line is because of the network latency time to access AWS cloud, although this can change based on the communication technologies used by vehicles including 3G, LTE, cellular or DSRC [9]. Clearly, this external policy engine does have some impact on the performance against no policy when used with number of vehicles. However, we believe when used in city wide scenario this time will be overshadowed by the notification time to all vehicles against a subset of vehicles provided by the policy evaluation engine. Our model and the use-case is focused to ensure service relevance to moving drivers on road which is well achieved even with a little tradeoff.

We understand that practical smart city transportation scenarios will have hundreds and thousands of moving cars (and other connected entities) associated to cloud (or fog infrastructures) and interacting. Although a detailed performance evaluation is eventually necessary by simulating large set of vehicles, we believe that our proof of concept in AWS is to showcase the practical viability and use of fine grained attribute based security policies in context of smart cars ecosystem, without the need to capture large set of data points from real world traffic scenarios spread across wide geographic area and sizable on-road moving vehicles. Such scaled setting will only stress the entire system without reflecting any change in security policy evaluation. We consider more detailed performance analysis as an extension to this work.

6 SUMMARY

This research work presents a fine-grained attribute-based access control model for time-sensitive and location-centric smart cars ecosystem. Our model introduces the novel notion of dynamic groups in relation to connected vehicles and emphasizes its relevance in this context. Besides considering system wide authorization policies, this model also supports personal preference policies for different users, which is required in today's privacy conscious world. Several real world use-cases are discussed and a proof of concept implementation of our CV-ABAC$_G$ model is shown in Amazon Web Services (AWS) cloud platform. This implementation demonstrates how moving vehicles can be dynamically assigned to location and sub-groups defined in the system based on the current GPS coordinates, vehicle-type and other attributes, besides the use of attribute based security policies in distributed and mobile connected cars ecosystem. Performance has been evaluated against time taken to determine activity access control decision when groups and ABAC policies are used against when no security policies are available. We envision to extend our model to introduce in-vehicle security and built risk aware trust-based models for smart vehicles environment. Also a more detailed evaluation and performance analysis of this model is needed to cover large set of moving vehicles, which is an enhancement to this work. Further, location privacy preserving approaches such as homomorphic encryption and other anonymity techniques can be used to complement and extend our model which can mitigate location sharing concerns without effecting its advantages and application. V2X trusted DSRC communication and privacy concerns also need further investigation, which we plan to explore as part of our future work in intelligent transportation.

ACKNOWLEDGEMENTS

REFERENCES

[1] 2014. *Connected Vehicles and Your Privacy.* https://www.its.dot.gov/factsheets/pdf/Privacy_factsheet.pdf
[2] 2017. *2017 Roundup Of Internet Of Things Forecasts.* https://www.forbes.com/sites/louiscolumbus/2017/12/10/2017-roundup-of-internet-of-things-forecasts/#67005b6a1480 [Online; Accessed: 2018-05-03].
[3] 2018. *AWS.* https://aws.amazon.com/ [Online; Accessed: 2018-05-09].
[4] 2018. *AWS Greengrass.* https://aws.amazon.com/greengrass/ [Online; Accessed: 2018-05-27].
[5] 2018. *AWS-IoT.* https://aws.amazon.com/iot/ [Online; Accessed: 2018-05-09].
[6] 2018. *AWS Lambda.* https://aws.amazon.com/lambda/ [Online; Accessed: 2018-05-20].
[7] 2018. *AWS SDK for Python (Boto3).* https://aws.amazon.com/sdk-for-python/ [Online; Accessed: 2018-05-23].
[8] 2018. *AWS Simple Notification Service.* https://aws.amazon.com/sns/ [Online; Accessed: 2018-05-20].
[9] 2018. *DSRC.* https://en.wikipedia.org/wiki/Dedicated_short-range_communications [Online; Accessed: 2018-08-07].
[10] 2018. *Google Maps Platform.* https://cloud.google.com/maps-platform/ [Online; Accessed: 2018-05-09].
[11] 2018. *Vehicular ad hoc networks.* https://en.wikipedia.org/wiki/Vehicular_ad_hoc_network [Online; Accessed: 2018-05-30].
[12] M. Aazam and et al. 2014. Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved. In *Proc. of IBCAST.* 414–419.
[13] A. Al-Fuqaha and et al. 2015. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Comm. Surveys & Tutorials* (2015), 2347–2376.
[14] Asma Alshehri and Ravi Sandhu. 2016. Access control models for cloud-enabled internet of things: A proposed architecture and research agenda. In *Proc. of IEEE CIC.* 530–538.

[15] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The internet of things: A survey. *Computer networks* 54, 15 (2010), 2787–2805.

[16] Jim Barbaresso and et al. 2014. USDOT's Intelligent Transportation Systems ITS Strategic Plan 2015- 2019. (2014).

[17] S. Bhatt, F. Patwa, and R. Sandhu. 2017. An Access Control Framework for Cloud-Enabled Wearable Internet of Things. In *Proc. of IEEE CIC*. 328–338.

[18] Smriti Bhatt, Farhan Patwa, and Ravi Sandhu. 2017. Access Control Model for AWS Internet of Things. In *Proc. of NSS*. Springer, 721–736.

[19] A. Botta, W. de Donato, V. Persico, and A. Pescapȧ. 2014. On the Integration of Cloud Computing and Internet of Things. In *Proc. of IEEE FiCLOUD*. 23–30.

[20] Mohamed Eltoweissy and et al. 2010. Towards Autonomous Vehicular Clouds. In *Ad Hoc Networks*. Springer, 1–16.

[21] ENISA. 2017. *Cyber Security and Resilience of smart cars: Good practices and recommendations*. https://www.enisa.europa.eu/publications/cyber-security-and-resilience-of-smart-cars [Online; Accessed: 2018-01-27].

[22] David F Ferraiolo, Ravi Sandhu, Serban Gavrila, D Richard Kuhn, and Ramaswamy Chandramouli. 2001. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)* 4, 3 (2001), 224–274.

[23] US GAO. 2016, March. Vehicle Cybersecurity . *GAO-16-350* (2016, March). https://www.gao.gov/assets/680/676064.pdf

[24] M. Gerla, E. Lee, G. Pau, and U. Lee. 2014. Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In *Proc. of IEEE WF-IoT*. 241–246.

[25] J. Gubbi and et al. 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems* 29, 7 (2013), 1645–1660.

[26] M. Gupta and et al. 2017. Multi-Layer Authorization Framework for a Representative Hadoop Ecosystem Deployment. In *Proc. of ACM SACMAT*. 183–190.

[27] Maanak Gupta, Farhan Patwa, and Ravi Sandhu. 2017. Object-Tagged RBAC Model for the Hadoop Ecosystem. In *Proc. of DBSec*. Springer, 63–81.

[28] Maanak Gupta, Farhan Patwa, and Ravi Sandhu. 2017. POSTER: Access control model for the Hadoop Ecosystem. In *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies*. ACM, 125–127.

[29] Maanak Gupta, Farhan Patwa, and Ravi Sandhu. 2018. An Attribute-Based Access Control Model for Secure Big Data Processing in Hadoop Ecosystem. In *Proc. of the Third ACM Workshop on Attribute-Based Access Control*. 13–24.

[30] Maanak Gupta and Ravi Sandhu. 2016. The GURA$_G$ Administrative Model for User and Group Attribute Assignment. In *Proc. of NSS*. Springer, 318–332.

[31] Maanak Gupta and Ravi Sandhu. 2018. Authorization Framework for Secure Cloud Assisted Connected Cars and Vehicular Internet of Things. In *Proc. of ACM SACMAT*. 193–204.

[32] Maanak Gupta and Ravi Sandhu. 2018. POSTER: Access Control Needs in Smart Cars. https://www.ieee-security.org/TC/SP2018/poster-abstracts/oakland2018-paper26-poster-abstract.pdf. [Online; Accessed: 2018-10-04].

[33] Per Hallgren, Martin Ochoa, and Andrei Sabelfeld. 2015. Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol. In *Privacy, Security and Trust (PST), 2015 13th Annual Conference on*. IEEE, 1–6.

[34] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen

Scarfone. 2014. Guide to attribute based access control (ABAC) definition and considerations. *NIST Special Publication* 800-162 (2014).

[35] Vincent C Hu, D Richard Kuhn, and David F Ferraiolo. 2015. Attribute-based access control. *IEEE Computer* 2 (2015), 85–88.

[36] Rasheed Hussain and et al. 2012. Rethinking vehicular communications: Merging VANET with cloud computing. In *Proc. of IEEE CloudCom*. 606–609.

[37] Xin Jin, Ram Krishnan, and Ravi Sandhu. 2012. A unified attribute-based access control model covering DAC, MAC and RBAC. In *DBSec*. Springer, 41–55.

[38] R. Lea and M. Blackstock. 2014. City Hub: A Cloud-Based IoT Platform for Smart Cities. In *Proc. of IEEE CloudCom*. 799–804.

[39] Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Michael Hamburg, Dan Boneh, et al. 2011. Location Privacy via Private Proximity Testing.. In *NDSS*, Vol. 11.

[40] NHTSA. 2016. NHTSA and Vehicle CyberSecurity. *NHTSA Report* (2016).

[41] NHTSA. 2016, October. Cybersecurity Best Practices for Modern Vehicles. *NHTSA Report No. DOT HS 812 333* (2016, October).

[42] M. Nitti and et al. 2016. The virtual object as a major element of the internet of things: a survey. *IEEE Comm. Surveys & Tutorials* (2016), 1228–1240.

[43] Stephan Olariu and et al. 2011. Taking VANET to the clouds. *International Journal of Pervasive Computing and Communications* 7, 1 (2011), 7–21.

[44] Jaehong Park, Ravi Sandhu, and Yuan Cheng. 2011. Acon: Activity-centric access control for social computing. In *Proc. of IEEE ARES*. 242–247.

[45] Jaehong Park, Ravi Sandhu, and Yuan Cheng. 2011. A user-activity-centric framework for access control in online social networks. *IEEE Internet Computing* 15, 5 (2011), 62–65.

[46] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. 1996. Role-based access control models. *Computer* 29, 2 (1996), 38–47.

[47] Ravi S Sandhu and Pierangela Samarati. 1994. Access control: principle and practice. *IEEE communications magazine* 32, 9 (1994), 40–48.

[48] Daniel Servos and Sylvia L Osborn. 2014. HGABAC: Towards a Formal Model of Hierarchical Attribute-Based Access Control. In *International Symposium on Foundations and Practice of Security*. Springer, 187–204.

[49] European Union. 2017. *Certificate Policy for Deployment and Operation of European Cooperative Intelligent Transport Systems (C-ITS)*. https://ec.europa.eu/transport/sites/transport/files/c-its_certificate_policy_release_1.pdf

[50] European Union. 2017. *Security Policy & Governance Framework for Deployment and Operation of European Cooperative Intelligent Transport Systems (C-ITS)*. https://ec.europa.eu/transport/sites/transport/files/c-its_security_policy_release_1.pdf

[51] USAToday. 2017. *Chinese group hacks a Tesla for the second year in a row*.

[52] USDOT. 2016. *Securty Credential Management System*. https://www.its.dot.gov/resources/scms.htm [Online; Accessed: 2018-01-13].

[53] Md Whaiduzzaman and et al. 2014. A survey on vehicular cloud computing. *Journal of Network and Computer Applications* 40 (2014), 325–344.

[54] Wired. 2015. *Hackers Remotely Kill a Jeep on the Highway-With Me in It*.

[55] Ge Zhong, Ian Goldberg, and Urs Hengartner. 2007. Louis, lester and pierre: Three protocols for location privacy. In *International Workshop on Privacy Enhancing Technologies*. Springer, 62–76.