



A Composition Algebra for Decentralized Enforcement of Access Control Policies with an Application to Vehicular Networks

Peter Amthor¹ ^a and René Gorges² ^b

¹Technische Universität Ilmenau, Ilmenau, Germany

²Friedrich Schiller University Jena, Jena, Germany

Keywords: Security Policy, Composition, Access Control, ABAC, Boolean Algebra, Vehicular Network, Smart Vehicles.

Abstract: Highly volatile and open distributed systems typically incorporate a significant amount of secure interactions between autonomous agents. This is especially true for vehicular networks, where smart or autonomous vehicles rely on information shared with each other or traffic infrastructure. However, controlling such decentralized interaction with respect to security restrictions requires a common definition of a temporary *composite policy*. As a first step towards this goal, this paper presents ACCA, a lightweight extension of Boolean algebra which allows to precisely specify how access control policies should be composed. It enables to build vehicular network systems that retain independence and autonomy of their participants while reducing the amount of communication about policy knowledge. An implementation of a simulation prototype of ACCA serves as a first, promising step towards tailoring the compositional semantics to specific use cases.

1 INTRODUCTION

Access control (AC) in open, distributed systems critically relies on the trustworthy and correct enforcement of individual rules about assets (access control policies). Such rules, in common terminology, precisely specify a mandatory condition to allow a subject (accessing IT activity) any operation (such as a remote procedure call) on an object (some resource abstraction). Especially in volatile, largely dynamic scenarios such as in ubiquitous and mobile computing or wireless networking, these policies are specified within *domains*: subjects that exact (physically or legally) control over objects and may provide other subjects access to them by the means of specific operations. As a whole, systems incorporating access control logic extending over multiple such domain-specific policies are logically considered a *composite policy group* (or just a group) (Kühnhauser, 1999).

In practice however, correctly evaluating access control rules in such scenarios is significantly complicated by three observations:


1. spontaneous formation and mutation of a group w. r. t. its member domains,


2. incomplete knowledge of each policy-enforcing domain about policies defined in a foreign domain,
3. uncertainty about formal correctness assumptions that any policy should satisfy from their domain's point of interest.

They become especially obvious in the communication between partially or fully autonomous cars or smart vehicles (SVs). Applications where such vehicles rely on information exchange and interoperation with other nearby vehicles and even traffic infrastructure are commonly summarized under the term vehicle-to-everything (V2X) communication. Facing mission-critical security and safety requirements, these distributed systems heavily rely on a sound and correct AC policy enforcement; at the same time, they illustrate the impact of two of the above-mentioned complications:

(1.) Groups are highly volatile in both a spatial and a temporal sense. This renders both the properties (such as physical location or speed) and the mere availability (presence in the system) of physically backed entities such as vehicles or their occupants highly dynamic.

(2.) Following from the previous observation but also from limited communication volume over the available infrastructure (such as vehicular ad-hoc net-

^a  <https://orcid.org/0000-0001-7711-4450>

^b  <https://orcid.org/0009-0001-3525-8445>

works (VANETs) (Hussein et al., 2022; Sharma and Kaul, 2021)), domains generally cannot make access decisions about shared resources (such as a dashboard camera media stream) based on complete knowledge of all group members and their respective local policy. As an example, streaming a camera image to another vehicle or a roadside unit (RSU) for traffic surveillance would be allowed from the camera’s vehicle owner’s perspective, but not from the perspective of another vehicle whose license plate or occupants’ faces would be in that footage. While the group policy could be built rather intuitively by conjunction of both vehicles’ policy decisions, in practice the access domain (with the camera on board) would need to first issue a secondary request to all other domains of possibly affected nearby vehicles.

Concerning the third observation, it should be noted that proving policy correctness is a problem not restricted to the scenarios characterized above. Nevertheless, when policy enforcement is done in a decentralized manner – in contrast to a centralized, non-circumventable, well-protected reference monitor (Anderson, 1972) as e. g. in operating systems – specific correctness properties must be demonstrable over all members of a group. In spite of a plethora of AC models tailored to analyzing and proving such properties, no widely accepted formalism exists for this purpose in the context of open, distributed systems and especially the V2X domain.

This problem field motivates our following contributions on fundamental AC policy composition logic:

- definition of a V2X system model and, based on it, a precise problem statement adaptable to other applications that share the above observations;
- design and definition of *ACCA* (Access Control Composition Algebra), a composition algebra for dynamic attribute-based access control (ABAC) models written in the DABAC modeling scheme (Schlegel and Amthor, 2023) that addresses these observations;
- a prototypical implementation of *ACCA* as a policy composition simulator, which allows for a first understanding of runtime behavior and possible applications of *ACCA* under different real-world scenarios.

The remainder of this paper is organized as follows: In § 2, we substantiate the policy composition problem in V2X applications with a motivating scenario, which is then generalized to characteristics of the underlying system (§ 3) and of the precise problem addressed (§ 4). We then give a brief overview of dynamic attribute-based access control (DABAC), an attribute-focused modeling scheme to describe AC

policy logic in § 5, and the actual algebra definition including a formal specification of composition operators in § 6. As a first step towards feasibility assessment for V2X applications, § 7 provides simulation results on a prototypical implementation using a virtual network environment. We review relevant related literature in § 8 and conclude with § 9.

2 V2X SCENARIO

In the following section we clarify our problem statement in the context of V2X communication. It should be noted that, after a brief review of background terminology, the presented scenario is just one specific example from a plethora of use cases in the literature. We will discuss the running use case in this paper in detail (§ 2.3), but also hint at other scenarios equally relevant to the policy composition problem.

2.1 Background

The term V2X (vehicle-to-everything) describes technologies which enable vehicles such as cars, trains or airplanes to communicate with each other as well as with road infrastructure and pedestrians (Arena and Pau, 2019; Mazzola et al., 2015; Hussein et al., 2022). These smart vehicles (SVs) are equipped with a variety of communication devices as well as sensors, e. g. cameras. Naturally, all of which manage data that potentially needs to be protected by an AC policy.

A common approach to integrating SVs into an architecture which supports V2X communication and its applications is vehicular edge computing (VEC) (Meneguet et al., 2021). This architecture puts RSUs, i. e. stationary communication units like traffic lights, at the edge of a cloud network. These communicate with, on one side, the mobile SVs and on the other side with cloud-based services (Zhou et al., 2022; Yang et al., 2022). The motivation for this design is the limited computational power and storage capacities of the SVs as well as the RSUs. SVs delegate computationally or storage-wise intensive tasks to RSUs, which in turn may delegate them to the cloud for more elaborate tasks, e. g. such involving augmented reality. Additionally, the cloud can utilize a global view of the vehicles’ area (Liu et al., 2020).

2.2 Motivating Scenario

From several real V2X applications described in the literature (Hussein et al., 2022; Alsarra et al., 2019), consider the following scenario to exemplify the necessity of multi-stakeholders policy composition. The

scenario focuses on coordinating regional traffic following a large-scale natural disaster, e. g. an earthquake. Disaster relief and emergency services typically rely on any real-time source of information, including data provided by private SVs such as camera images and location data. The policy involved here is from the family of break-glass AC policies (Marinovic et al., 2011): Access to sensitive data should only be permitted to specific subjects (or rather, depending on the model, their roles) that are principals of emergency services and only in presence of an emergency; at the same time, it must not be deniable either under such conditions.¹

2.3 Running Use Case

More precisely, the above scenario can be generalized to the following use case. Given three classes of parties in an urban transportation setting:

- **I**: urban traffic infrastructure, which provides services such as parking lot allocation and navigation through mostly immovable RSUs,
- **V**: private and commercial vehicles,
- **E**: emergency services vehicles, such as firefighters or ambulances.

Each individual stakeholder participating in this setting would belong to exactly one of these classes and thus form a domain on its own. While each individual vehicle in the class **V** forms its own domain d_i , we assume exactly one domain d_I and d_E for the other two, respectively.

Assume that, within a spatially delimited area (e. g. bounded by wireless communication range) and at a specific point in time, a vehicle v from **V**, which is equipped with various sensors such as cameras, records data which can be helpful for parties from **E** by enabling them a quick and appropriate response to a specific emergency situation. If the relevant data can be transmitted to those parties (possibly through the urban infrastructure), it should be. Consequently, parties from **E** may request access to data within v 's domain d_v , which, if granted, would allow the transfer of (possibly sensitive) data from d_v to d_E .

This use case includes two stakeholders with opposing interests: The owners and users of SVs want to share as little sensitive information as possible, but the emergency personnel has to use their sensor data for the purpose of public safety, e. g. to aid a coordinated evacuation and thus save human lives. Com-

¹The post-operational obligations typical for break-glass policies – e. g. logging emergency vehicle UIDs and archiving their camera footage – are relevant in practice, but ignored in the discussion here for the sake of clarity.

binning these two goals requires the definition of an access control policy which is a combination of both stakeholders' individual policies. Notably, this combined policy will make decisions depending on (1) external circumstances, e. g. emergency notifications received by cell broadcast infrastructure, and (2) subject metadata, as the aforementioned (cryptographically authenticated) emergency role certificate.

The resulting policy is an example for a dynamic *composite policy*, whereas the individual policies are called *component policies*.

2.4 Other Scenarios

Apart from the scenario outlined above, which we will refer to throughout the rest of this paper, policy composition problems are common in a number of other V2X scenarios. As an example, use of a civilian's dashboard camera data to track down criminals as described in (Alsarra et al., 2019) comes with the same implications as the disaster relief scenario. Another scenario is outlined in (Hussein et al., 2022): Consider a city offering an electronic parking guidance system. A driver looking for a parking spot can let their car query the system for information on nearby parking spots. The parking guidance system would then request necessary data from the vehicle, for example its dimensions, mass, destination and potentially license number. This may be a problematic access though, e. g. in case the vehicle is commercially operated and the owner does not consent to this data to be passed to the system. When, on the other hand, the driver's highest priority is to find a parking spot and to arrive at a business appointment on time, multiple stakeholders with opposing interests can be found in this scenario. A solution to this conflict by composing their access control policies needs to reflect these interests.

3 SYSTEM MODEL

We describe the AC characteristics of an open, heterogeneous and spatially dynamic distributed system based on VEC infrastructure by the following model.

Each stakeholder (such as drivers and owners of SVs – who might be different – in the scenario of § 2.3) forms a *domain*. It can be intuitively understood as a collection of subjects and objects that are spatially bound to each other and thus share co-located (e. g. memory-based) communication infrastructure. More formally, we define a domain $d = \langle E_d, Op_d \rangle$ as a tuple of *entities* from a set E_d and *operations* from a set Op_d . Entities might be principals

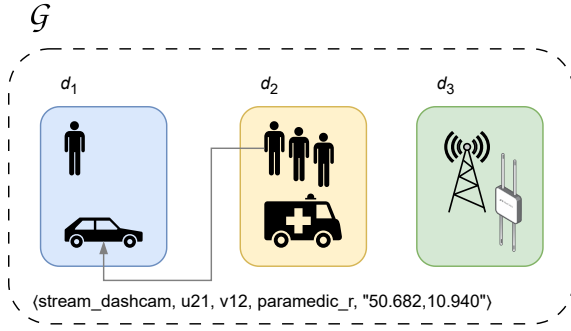


Figure 1: Example scenario for a group \mathcal{G} with three domains $d_{1,2,3}$, one out of either private vehicles (blue), emergency services (yellow), and traffic infrastructure (green).

(i. e. abstractions of human users), subjects acting on their behalf, or technical or logical assets or other objects in the scope of d .² In the running use case, these include e. g. emergency personnel, SV users, passive occupants of emergency vehicles etc. (cf. Fig. 1). Operations specified in Op_d identify any possible technical activities that involve entities in E_d . This includes operations *known to affect* the domain’s entities, but not necessarily implemented inside the domain.³ All domains whose entities, for a specific point in time, might interact with each other within the same communication infrastructure (such as a vehicular network), form a *group* \mathcal{G} . We define $E_{\mathcal{G}} = \bigcup_{d \in \mathcal{G}} E_d$ as the group’s entity set and $Op_{\mathcal{G}} = \bigcup_{d \in \mathcal{G}} Op_d$ as its set of operations. In general and for two subsequent, temporally discrete points in time (*states* of the system) \mathcal{G} and \mathcal{G}' , we assume that $\mathcal{G} \neq \mathcal{G}'$. This models dynamic group formation and spontaneous mutation (i. e. SVs joining or leaving the group at any time).

An access request is a tuple $\langle op, a_1, \dots, a_n \rangle$, describing an operation $op \in Op_{\mathcal{G}}$ that should be performed for a vector $[a_i]_{i=1}^n$ of parameters (specific to each op), which might be unified with identifiers from $E_{\mathcal{G}}$ or any operation-specific value. Notably, this includes – additional to unique identifiers (UIDs) – attributes of entities or the physical environment. Fig. 1 shows an exemplary access request with $op = \text{stream_dashcam}$ and various parameters: UID of the calling entity $u21$, UID of d_1 ’s dashboard camera controller $v12$, and two more attribute values to describe the caller’s role (paramedic_r) and GPS coordinates as environmental context of the request.

A self-contained specification of AC rules, which

²Note that we intentionally use “entity” to subsume the intuitive, yet formally more restrictive distinction between subjects (always active) and objects (always passive).

³Our earlier example of recording video footage of an SV, which is obviously not recorded from within its domain, may require consent based on its domain’s policy.

completely defines for any possible access request $\langle op_d, a_1, \dots, a_n \rangle, op_d \in Op_d$ if the modeled operation is allowed to be performed, is called d ’s *domain policy* P_d . Analogously, $P_{\mathcal{G}}$ for $Op_{\mathcal{G}}$ is called the *composite policy* of group \mathcal{G} . Finally, an *access control function* (ACF) f_P is a mapping defined as $\langle op, a_1, \dots, a_n \rangle \mapsto \text{true}$ iff operation op is allowed with parameters $[a_1, \dots, a_n]$ by policy P , which might be a domain policy or a composite policy. Any AC decision modeled by f_P requires, from a subject’s perspective, only operation-specific parameters, but no knowledge about the (current state of the) group and its composite policy. This enables a middleware framework to transparently implement group management and composite policy evaluation.

4 PROBLEM STATEMENT

We now describe the general logical composition problem for spatially and/or temporally independent AC domains which are autonomous in their respective local policies. It is characterized by the following properties:

Dynamic Group A group is dynamically formed and mutated by adding or removing domains. Policy composition must be done in an ad-hoc manner.

Transparent Enforcement Policy composition is transparent for all entities to allow for group-agnostic domain interfaces.⁴

Policy composition can be recursive, which guarantees the same AC semantics for requests to both a local and the composite policy.

Domain Autonomy For any access request against the composite policy, only such component policies whose domains include entities from the request should be taken into account. This ensures that any request whose entities are all local to one domain is handled identically by the composite policy and that domain’s component policy; at the same time, it prevents completely unrelated component policies from making non-meaningful decisions.

Policy Model A generic⁵ formal language is agreed upon by all domains, which is used to precisely express and possibly verify component policies as well as the composite policy.

⁴This enables e. g. general-purpose application software to act as an entity in the composite policy.

⁵In a sense that all possible domains of some application, such as V2X, can express their respective local policies.

For the design of *ACCA*, we make the following underlying assumptions. Their establishment by communication infrastructure and hard- and software stack in each participating domain is regarded subject to complementary technology:

First, for any access request against the composite policy, all domains which include entities from the request must enforce the composite policy’s decision. This ensures that no domain might ignore the composite policy and can be ensured e.g. through a trusted execution environment (TEE) infrastructure. Second, entity UIDs and metadata for attribution of entities are authentic. This can be cryptographically ensured. Third, communication between all domains is non-repudiable and never fails, which can be addressed by a trusted, fault-tolerant middleware. It should be mentioned that all these complementary components will inevitably contribute to the size of a system’s trusted computing base (TCB).

5 ACCESS CONTROL POLICY MODEL

As already motivated before, both the component policies and a composite policy should be open to formal analyses of correctness properties. Amongst the most challenging of these are *dynamic* properties, i.e. state reachability problems such as HRU safety (Harrison et al., 1975; Sandhu, 1992), workflow satisfiability (Khan and Fong, 2012), or delegation of trust (Li et al., 2005). So far, however, open distributed systems largely rely on policy specification languages such as XACML (OASIS, 2013) or NGAC (INCITS, 2013; Ferraiolo et al., 2016) which are based on ABAC semantics: since attributed-based policies can specify rules without any prior knowledge about subjects, objects, or other entities present during enforcement time, they are by design a natural choice for V2X access control (Ashutosh et al., 2023; Gupta et al., 2019). Unfortunately, none of them is formally accessible to the analysis of dynamic correctness properties.

To formally express both the knowledge and compositional semantics of ABAC policies, we therefore choose a modeling scheme which is (1) flexible enough to allow for diverse attribution semantics (cf. § 2), (2) applicable to V2X as our target application domain, and (3) accessible to formal verification of dynamic properties. All these requirements are satisfied by DABAC, which we will outline in the remainder of this section.

5.1 DABAC Modeling Scheme

DABAC is a modeling scheme for ABAC policies proposed in (Schlegel and Amthor, 2021). Its focus is to reason about dynamic changes of a system controlled by an ABAC policy. We will now briefly reiterate those aspects of DABAC relevant to this work.

Definition 1 (DABAC Policy). A **DABAC Policy** with indirection degree i_{\max} is a tuple $\langle EN, EN^{\text{ext}}, AV, AV^{\text{ext}}, AA, AA^{\text{ext}}, AR \rangle$, where

- $EN = \{E_1, \dots, E_n\}$ is a family of entity sets,
- $AV = \bigcup_{i=0}^{i_{\max}} AV_i$ are families of attribute value sets, $AV_i = \{V_1, \dots, V_{n_i}\}$,
- $AA = \bigcup_{i=0}^{i_{\max}} AA_i$ are families of attribute association sets, where $AA_0 = \{att : E \rightarrow V'_0 \mid E \in EN\}$ and $AA_i = \{att : V_{i-1} \rightarrow V'_i \mid V_{i-1} \in AV_{i-1}\}$ such that V'_i is either a set in AV_i or a power set of a set in AV_i ,
- $Perm$ is a set of access permissions,
- $AR = \{auth_p : [v_1, \dots, v_{n_p}] \rightarrow \mathbb{B}, n_p \in \mathbb{N}\}$ is a set of authorization predicates (or “access rules”), exactly one for each permission $p \in Perm$, which map a finite set of arguments v_k to a Boolean access decision such that $\forall k \in [1, n_p] : \exists V_k \in AV \cup AV^{\text{ext}} \wedge (v_k \in V_k \vee v_k \in 2^{V_k})$,

and $0 \leq i \leq i_{\max}$. Analogous to above, EN^{ext} , AV^{ext} and AA^{ext} denote *policy-external* entity sets, attribute value sets, and attribute association sets.

DABAC distinguishes between entities, attribute values and attribute associations which are physically within the local domain (*internal*) and those which are not. The latter are termed *external* and cannot be influenced from inside the local domain. Furthermore, it is assumed that this distinction yields disjoint sets of internal and external entities, attribute values and attribute associations respectively.

In definition 1, the *indirection degree* of attribute values is used. This term describes a number $i \in \mathbb{N}$ which makes indirect attribution possible. To understand the mindset behind this design, consider attributes which are more complex than an atomic value, like a record containing multiple pieces of information on a car, including for example the year of construction, vehicle identification number, fuel type and more. It might not be desirable to split this record into the individual values that compose it – especially as there might be dependencies among them – but rather treat it as one structured value. To achieve this, such records can be modeled as attributes which themselves have attributes, namely the elements of the respective record. In a DABAC policy, this is done through indirect attribution. Attribute values with indirection degree $i = 0$ are elements from or subsets

of a set from AV_0 or AV_0^{ext} and hence atomic values. A value $i > 0$ means that the corresponding attribute value is composed of attribute values of the indirection degree $i - 1$, that is a structured value. In order to determine the value of an attribute with indirection degree $i > 0$, a mapping from AA_i or AA_i^{ext} has to be evaluated. It should be noted that attribute values of any indirection degree can be a single or a set of (possibly structured) values.

Finally, the set AR determines the rules by which accesses are determined to be legal or not. Its elements are mappings $auth_p$ for every permission p from the set $Perm$ of all permissions, which map n_p attribute values to a Boolean value. Given the attribute values $[v_i]_{i=1}^{n_p}$, for example through an entity's access request and the physical environment, the permission $p \in Perm$ is granted if and only if $auth_p([v_i]_{i=1}^{n_p}) = \text{true}$. In order to execute an operation, all permissions required for that operation must be granted.

Given an operation $op \in OP$ and parameters $[a_i]_{i=1}^{n_{op}}$ required for it, the ACF of a DABAC policy should decide whether or not the thereby requested access is legal, where OP is the set of all operations and n_{op} denotes the number of parameters required for op . For this decision to be made, it is necessary to evaluate $auth_p \in AR$ for every permission $p \in Perm$ required to execute op . However, this requires to map the parameters $[a_i]_{i=1}^{n_{op}}$ to according parameters of $auth_p$. This is achieved by mappings which are part of the previously mentioned automaton's state transition function, which is not shown here. We use the notation $\langle a_1, \dots, a_{n_{op}} \rangle \models op.PRE$, similar to (Schlegel and Amthor, 2021), which describes a Boolean formula with variables $X_1, \dots, X_{m_{op}}$ that is satisfied by an assignment $\forall i \in [1, m_{op}] : \exists j \in [1, n_{op}] : X_i := a_j$. Any such formula must be a conjunction of those $auth_p$ which have to be satisfied in order for op to be permitted. Finally, both the operation and its parameters are the input to the ACF, where Σ is defined as the set of all possible such inputs. A formal definition of Σ and $op.PRE$ is given in (Schlegel and Amthor, 2021).

Using these notations, the ACF for a DABAC model can be defined.

Definition 2 (DABAC ACF). The access control function of a DABAC model for a policy P is a function $f_P : \Sigma \rightarrow \mathbb{B}$, where

$$f_P(op, [a_i]_{i=1}^{n_{op}}) = \text{true} \Leftrightarrow op.PRE.$$

5.2 Example Policy

Given the use case described in § 2.3, we can now define a DABAC model to represent the interests of car owners and emergency forces when accessing a camera (operation $accessCam$):

- $EN = \{C, U\}$, where C is a set of camera identifiers and U is a set of user identifiers,
- $AV_0 = \{L, \mathbb{B}\}$ where $L = [-90, 90] \times [-180, 180]$ is a set of geographical locations and \mathbb{B} is the set of Boolean values,
- $AV_1 = \{WL\}$, where $WL = \{0, 1, \dots, 5\}$ is the set of possible values for a numerical system of warning levels, where 0 indicates no warning and 5 indicates the highest possible warning level,
- $AA_0 = \{att_{CL}, att_{EF}\}$, where $att_{CL} : C \rightarrow L$ assigns a geographical location to every camera and $att_{EF} : U \rightarrow \mathbb{B}$ assigns every user a Boolean value indicating whether this user represents an emergency force and
- $AA_1 = \{att_{LWL}\}$, where $att_{LWL} : L \rightarrow WL$ assigns every location a warning level.

Let P_1 be the policy of the car owner Alice. Specifically, we define $C = \{cAlice\}$, $U = \{uAlice, uFireTruck\}$, $att_{CL}(cAlice) = \langle 50.41, 10.55 \rangle$, $att_{EF}(uAlice) = \text{false}$, $att_{EF}(uFireTruck) = \text{true}$, $att_{LWL}(\langle 50.41, 10.55 \rangle) = 4$ and $accessCam.PRE(u \in U, c \in C) \Leftrightarrow u = uAlice \wedge c = cAlice$ for P_1 .⁶ As can be seen, P_1 only allows camera access to the camera $cAlice$ by the user $uAlice$.

Analogously, we define a policy P_2 for a fire truck which is structurally identical to P_1 . For P_2 , use the same sets of entities, attribute values and attributions. Additionally, define the access rules $auth_{isCritical}$ and $auth_{isEF}$ as $auth_{isCritical}(wl \in WL) \Leftrightarrow wl \geq 4$ and $auth_{isEF}(u \in U) \Leftrightarrow att_{EF}(u)$. Furthermore, define $accessCam$'s precondition by $accessCam.PRE(u \in U, c \in C) \Leftrightarrow auth_{isEF}(u) \wedge auth_{isCritical}(att_{LWL}(att_{CL}(c)))$, i. e. P_2 allows a user u to access a camera c iff u represents an emergency force and c is currently in a location with a critical warning level.

Following Def. 2, it can be seen that $f_{P_1}(accessCam, \langle uFireTruck, cAlice \rangle) \Leftrightarrow accessCam.PRE(uFireTruck, cAlice)$ which is false; i. e. Alice's policy doesn't allow the fire truck to access her camera. On the other hand, it holds that $f_{P_2}(accessCam, \langle uFireTruck, cAlice \rangle) \Leftrightarrow accessCam.PRE(uFireTruck, cAlice) \Leftrightarrow \text{true}$, i. e. P_2 would grant the fire truck access to Alice's camera.

In this scenario, it can be seen that Alice is currently in a location with a critical warning level, which means that the fire truck should be able to access Alice's camera data. To achieve this, P_1 and P_2 have to be composed.

⁶This is a formal simplification, as operation preconditions may only operate on attribute values. A necessary, additional layer of indirection was omitted here for brevity.

6 COMPOSITION ALGEBRA

Based on the DABAC model for AC policies introduced in the previous section, we define *ACCA* as an extension of Boolean algebra. We will outline the design requirements (§ 6.1) and the definition of composition operators (§ 6.2), and finally define an exemplary composition scenario based on the use case in § 2.3 (§ 6.3).

6.1 Algebra Design

We define the following logical requirements for a policy composition algebra. First, composition operations must be associative and commutative. Otherwise, the order in which policies enter a group would affect access decisions, which violates the assumption of dynamic group (re-) formation and group-transparent AC decision making (§ 4). Second, composite policies must be structurally identical to component policies to allow for recursive composition and unified AC evaluation interfaces.

Third, due to generally limited bandwidth of underlying communication infrastructure, the amount of information available about component policies is a limiting factor for policy composition. Generally, it is desirable to have as much information as possible available to define a proper DABAC policy as the result of the composition, but this has the drawback that it leads to a large amount of communication.

This is why *ACCA* is defined in three different tiers, according to different amounts of information which is available on the component policies. They can be described as follows:

- Tier 1.** Only the outputs of the component policies' ACFs are known.
- Tier 2.** The access rules and preconditions of operations of the component policies are known.
- Tier 3.** The entire component policies, that is all their components are known.

Additional to limited information, potentially flawed communication imposes another problem. In tier 1, for example, outputs of ACFs are known in the sense that they can be queried from the owner of their respective policy. However, due to errors in the communication – e. g. because the queried car just left communication reach – such queries may not be met by a proper response, which makes the queried values unavailable.

For the rest of this paper, the symbol \square denotes a value that is unavailable (not necessarily undefined). There are two ways to deal with unavailable values when composing policies: deny any access request

when at least one value necessary for the policy evaluation is unavailable, or ignore unavailable values.

Consider the conjunction of two policies, that is a composite policy which grants an access iff both component policies grant it (assuming both can make a meaningful decision). The above distinction leads to two different versions of the policy conjunction, which will be introduced as \wedge_M for making the availability of all necessary values mandatory and \wedge_D for disregarding unavailable values. Analogous, the operations \vee_M and \vee_D will be used for the disjunction of policies, where the disjunction of two policies is a composite policy which grants an access iff at least one of the component policies grant it (again assuming that both can make a meaningful decision).

These four operations lead to an algebra for composing policies which is more expressive than a composition using only the trivial extensions of the Boolean operators \wedge and \vee .

6.2 Composition Logic

As the three tiers differ in the amount of information available, different definitions have to be made for them; the basic idea is the same for all three, though.

Tier 1. For the tier 1 policy composition, two policies P_1 and P_2 with their respective ACF $f_i : \Sigma_i \rightarrow \mathbb{B} \cup \{\square\}$, $i \in \{1, 2\}$ are given. Let $f : \Sigma \rightarrow \mathbb{B} \cup \{\square\}$ be the ACF of the composite policy P , where $\Sigma = \Sigma_1 \cup \Sigma_2$. For the composition $P = P_1 \oplus_M P_2$, where $\oplus \in \{\wedge, \vee\}$, the value of $f(\sigma)$ for $\sigma \in \Sigma$ is given by algorithm 1.

```

if  $\sigma \in \Sigma_1 \cap \Sigma_2 \wedge \square \notin \{f_1(\sigma), f_2(\sigma)\}$  then
  | return  $f_1(\sigma) \oplus f_2(\sigma)$ 
else if  $\sigma \in \Sigma_1 \setminus \Sigma_2 \wedge f_1(\sigma) \neq \square$  then
  | return  $f_1(\sigma)$ 
else if  $\sigma \in \Sigma_2 \setminus \Sigma_1 \wedge f_2(\sigma) \neq \square$  then
  | return  $f_2(\sigma)$ 
else
  | return  $\square$ 
end
    
```

Algorithm 1: The specification of the ACF f of $P = P_1 \oplus_M P_2$ for $\oplus \in \{\wedge, \vee\}$ for the tier 1 composition.

As can be seen, the ACFs f_1 and f_2 are combined using the underlying Boolean operation if both policies should be considered (i. e. $\sigma \in \Sigma_1 \cap \Sigma_2$) and if neither ACF is currently unavailable. If only one policy should be considered, then only the value of its corresponding ACF is used, provided this value is not unavailable. If, however, the value of an ACF is required but not available, the new ACF f evaluates to \square . Note that returning \square here is necessary to make the composition operations associative.

```

if  $\sigma \in \Sigma_1 \cap \Sigma_2 \wedge \square \notin \{f_1(\sigma), f_2(\sigma)\}$  then
  | return  $f_1(\sigma) \oplus f_2(\sigma)$ 
else if  $f_1(\sigma) \neq \square \wedge (\sigma \in \Sigma_1 \setminus \Sigma_2 \vee \sigma \in$ 
  |  $\Sigma_1 \cap \Sigma_2 \wedge f_2(\sigma) = \square)$  then
  | return  $f_1(\sigma)$ 
else if  $f_2(\sigma) \neq \square \wedge (\sigma \in \Sigma_2 \setminus \Sigma_1 \vee \sigma \in$ 
  |  $\Sigma_1 \cap \Sigma_2 \wedge f_1(\sigma) = \square)$  then
  | return  $f_2(\sigma)$ 
else
  | return  $\square$ 
end

```

Algorithm 2: The specification of the ACF f of $P = P_1 \oplus_{\mathbb{D}} P_2$ for $\oplus \in \{\wedge, \vee\}$ for the tier 1 composition.

When computing $P = P_1 \oplus_{\mathbb{D}} P_2$ for $\oplus \in \{\wedge, \vee\}$, the value $f(\sigma)$ of P 's ACF is given by algorithm 2. The differences compared to algorithm 1 lie in the second and third if-statement. The new ACF f uses only one of the original policies' ACF (provided it is available) if either only one policy is concerned with the access request (as before) or both are, but one ACF value is currently unavailable.

The introduced definitions can only be meaningful if two assumptions about the policies P_1 and P_2 are met. First, all entities that may request access to any entity must be known by both policies. Consider a car whose owner's policy is P_1 and whose driver's policy is P_2 . If P_2 contains an entity which can request access to e. g. the car's camera data but which is not part of P_1 , any access request involving that entity cannot be an element of Σ_1 . Consequently, P_1 has no influence on the decision a policy composed of P_1 and P_2 may make, presented this access request. However, the car's owner should be involved in this decision. Hence, P_1 has to account for such entities.

Second, consider two policies P_1 and P_2 which both account for an entity which can be the object of an access request, but is external in exactly one of both, say P_2 . This means that the access to this entity is governed only by P_1 . In a composite policy $P = P_1 \oplus_{\alpha} P_2$ with $\alpha \in \{M, D\}$, the access to this entity should still be governed only by P_1 . Consequently, the ACF f_2 which P_2 provides for the composition has to return a value which is neutral to \oplus (i. e. the Boolean operation upon which \oplus_{α} is based) so that its decision has no effect on the value of P 's ACF.

Tier 2. For tier 2, the two policies are of the form $P_i = \langle AR_i \rangle, i \in \{1, 2\}$. For the policy composition, both policies' access rules and the preconditions of P_1 's and P_2 's operations need to be redefined to take into account the desired behaviour when encountering a \square -value. For this, we define $att(\square) = \square, auth_p(\square) = \square$ and $\square * a = a * \square = \square$ for all of

P_i 's attributes att , access rules $auth_p$ as well as for all operators $* \in \{+, -, \cdot, /, =, \in, \subseteq, \dots\}$, where $=$ is the comparison operators, and according operands a . Note that $* \notin \{\wedge, \vee, =\}$. For an access rule $auth_p \in AR_1 \cup AR_2$, define $auth'_p$ as the function which behaves exactly like $auth_p$, but returns \square if any of the values that arise during the evaluation of $auth_p$ is \square . Formally, if $auth_p$ is defined by the expression $\bigvee_{i=1}^n \bigwedge_{j=1}^{m_i} \phi_{i,j}$ in disjunctive normal form if some condition β is met and as \square otherwise, then $auth'_p$ returns the value of $auth_p$ if the condition $\beta \wedge \forall i, j : \phi_{i,j} \neq \square$ is met and \square otherwise. Analogously, for every $op \in OP_1 \cup OP_2$, the new precondition $op.PRE'$ is defined.

Then, given two policies $P_i = \langle AR_i \rangle$ with ACF domain Σ_i for $i \in \{1, 2\}$, define $\Sigma = \Sigma_1 \cup \Sigma_2$, $Perm = Perm_1 \cup Perm_2$, $AR = \bigsqcup_{k=1}^2 \{auth'_p \mid auth_p \in AR_k\}$ and $OP = OP_1 \cup OP_2$. Lastly, for every $op \in OP$, let $op.PRE$ be defined by algorithm 1, where Σ_i and f_i are replaced by OP_i and $op.PRE'_i$ respectively and σ is an element of $op.PRE'$'s domain. The policy $P = P_1 \oplus_M P_2$ for $\oplus \in \{\wedge, \vee\}$ is defined as the policy $P = \langle AR \rangle$ with the above defined sets of access rules AR , permissions $Perm$ and operations OP . Its ACF f is given by $f(op, [a_i]_{i=1}^{n_{op}}) = op.PRE([a_i]_{i=1}^{n_{op}})$.

For the composition $P_1 \oplus_{\mathbb{D}} P_2$, the access rules and operation preconditions have to be defined differently to disregard the \square -values that might be encountered during their evaluation. Given an expression $\bigvee_{i=1}^n \bigwedge_{j=1}^{m_i} \phi_{i,j}$ in disjunctive normal form, the components of every conjunction should be treated as true when they evaluate to \square , except for when all parts of a conjunction evaluate to \square , in which case the whole conjunction should be treated as false. This is realized by the expression $\psi = \bigvee_{i=1}^n \left(\bigwedge_{j=1}^{m_i} \phi_{i,j} \in \{\text{true}, \square\} \wedge \bigvee_{j=1}^{m_i} \phi_{i,j} \neq \square \right)$. Like before, let $auth_p$ be an access rule which is defined by the expression $\bigvee_{i=1}^n \bigwedge_{j=1}^{m_i} \phi_{i,j}$ in disjunctive normal form if some condition β is met and as \square otherwise. Then, the function $auth''_p$ evaluates to ψ if the condition $\beta \wedge \exists i, j : \phi_{i,j} \neq \square$ is met and to \square otherwise. Define $op.PRE''$ analogously for every operation $op \in OP_1 \cup OP_2$.

Then, the policy $P = P_1 \oplus_{\mathbb{D}} P_2$ for $\oplus \in \{\wedge, \vee\}$ is defined as the policy with the set of permissions $Perm = Perm_1 \cup Perm_2$, the set of operations $OP = OP_1 \cup OP_2$ and the set of access rules $AR = \bigsqcup_{k=1}^2 \{auth''_p \mid auth_p \in AR_k\}$. For every operation $op \in OP$, let $op.PRE$ be defined by algorithm 2, where Σ_i and f_i are replaced by OP_i and $op.PRE''_i$ for $i \in \{1, 2\}$ respectively and where σ is an element of $op.PRE''$'s domain.

Additional to the assumptions that were required

for tier 1, the tier 2 composition requires that every entity which may act as an object in an access request has to be included in both P_1 and P_2 . This is because then, the access rules $auth_p$ as well as the $op.PRE$ functions can return a meaningful value when presented arguments their respective policy did not originally account for.

Tier 3. The third tier composes two policies of which all components are known. Note that this means that no value will ever evaluate to \square , as the component policies are available locally.

For the tier 3 composition, the distinction of internal and external entities and attribute values through the sets EN^{ext} , AV^{ext} and AA^{ext} in DABAC as described in § 5.1 is problematic. Composing policies while keeping these sets means that the resulting composite policy's entities and attribute values are internal or external to either every or none of the component policies' domain. This, however, is not generic enough, as access decisions may incorporate knowledge on whether an entity or attribute value is internal or external to a component policy, which should be retained in the composite policy. Rather, it should be possible for entities and attribute values to be internal or external to only some of those domains, i. e. the information on externality from the component policies should be kept. To achieve this, we incorporate according attributions and access rules into the component policies, which makes the definition of EN^{ext} , AV^{ext} and AA^{ext} unnecessary. Consequently, we will not consider these sets in the following definition.

Let $P_1 = \langle EN_1, AV', AA', AR_1 \rangle$ and $P_2 = \langle EN_2, AV'', AA'', AR_2 \rangle$ be two structurally identical policies with $EN_k = \{E_{k,1}, \dots, E_{k,n}\}$ for all $k \in \{1, 2\}$ as well as $AV'_i = \{V'_{i,1}, \dots, V'_{i,n_i}\}$ and $AV''_i = \{V''_{i,1}, \dots, V''_{i,n_i}\}$ for all $0 \leq i \leq i_{max}$. Define the sets $EN = \{E_1, \dots, E_n\}$ with $E_j = E_{1,j} \cup E_{2,j}$ for all $1 \leq j \leq n$ and $AV_i = \{V_{i,1}, \dots, V_{i,n_i}\}$ with $V_{i,j} = V'_{i,j} \cup V''_{i,j}$ for all $1 \leq j \leq n_i$ and all $0 \leq i \leq i_{max}$. Furthermore, define $AA = \{att' \mid att \in AA'\} \cup \{att' \mid att \in AA''\}$, where att' is equal to att for all arguments from att 's domain and undefined for all other arguments. Then, for $\oplus_\alpha \in \{\wedge_M, \wedge_D, \vee_M, \vee_D\}$, the policy $P = P_1 \oplus_\alpha P_2 = \langle EN, AV, AA, AR \rangle$ is the policy with the above defined sets of entities, attribute values and attributions and the remaining components defined as in tier 2.

6.3 Example

Let P_1 and P_2 be the two policies defined in § 5.2. In the scenario (as described in § 2.3), policy P_2 should

be able to overrule P_1 's decision to deny the firetruck represented by $uFireTrack$ access to Alice's camera $cAlice$. To achieve this, it is reasonable to compute a composite policy P as $P = P_1 \vee_M P_2$. For tier 1, this means that f_P is defined by algorithm 1 with $\oplus = \vee$. Let $\sigma = \langle accessCam, [uFireTrack, cAlice] \rangle$ be an access request. As was shown in § 5.2, it is $f_{P_1}(\sigma) = \text{false}$ and $f_{P_2}(\sigma) = \text{true}$. Assuming that both values are available, it follows that $f_P(\sigma) = f_{P_1}(\sigma) \vee f_{P_2}(\sigma) = \text{true}$, i. e. the composite policy P grants the fire truck access to Alice's camera as policy P_2 grants it.

For the tier 2 composition, we modify the definition of $accessCam.PRE$ in P_1 to return $u == uAlice \wedge c == cAlice$ (now using $==$ as the comparison operator) iff $(u == uAlice) \neq \square$ and $(c == cAlice) \neq \square$ and \square otherwise. This new precondition will be called $accessCam.PRE'_1$. Analogously, we define $auth'_{isCritical}$, $auth'_{isEF}$ and $accessCam.PRE'_2$ for P_2 . Then, $P = P_1 \vee_M P_2$ is the policy with the set of access rules $AR = \{auth'_{isCritical}, auth'_{isEF}\}$ and the operation $accessCam$. When all values are available, all attributions and access rules of P are identical to the original attributions and access rules in P_1 and P_2 respectively. Furthermore, it is $accessCam.PRE = accessCam.PRE'_1 \vee accessCam.PRE'_2$, where $accessCam.PRE'_i$ is identical to the definition of $accessCam.PRE$ in P_i for $i \in \{1, 2\}$, again because all necessary values are assumed to be available. Then, it can be seen that

$$\begin{aligned} f_P(\sigma) &\Leftrightarrow accessCam.PRE(uFireTrack, cAlice) \\ &\Leftrightarrow accessCam.PRE'_1(uFireTrack, cAlice) \\ &\quad \vee accessCam.PRE'_2(uFireTrack, cAlice) \\ &\Leftrightarrow \text{false} \vee \text{true} \Leftrightarrow \text{true}. \end{aligned}$$

Again, P grants the fire truck access to Alice's cam.

The policy which is computed using tier 3 has identical attributions, access rules and operations (including their preconditions) as the result of the tier 2 composition. Additionally, a set EN and AV are defined. As the sets EN and AV of P_1 and P_2 are identical, the policy $P = P_1 \vee_M P_2$ also uses these sets. Putting together the example evaluation for the tier 2 composition as well as the evaluation of σ in P_1 and P_2 from § 5.2, it can be seen that P again grants the fire truck the desired access to Alice's camera.

7 EVALUATION

To analyze the feasibility of the proposed algebra in the scenario described in § 2, several runtime measurements were conducted on a prototype implemen-

tation of *ACCA*.⁷

For the measurements, a network of n hosts, each of which being connected to a single switch, is simulated using the software *mininet*. For that, a virtual machine image running Ubuntu 20.04.1 from the official *mininet* website with *mininet* preinstalled was set up in Oracle VM VirtualBox 7.0.8 and executed on a system with a 4-core AMD Ryzen 5 3500U CPU and 8 GB main memory running Pop!_OS 22.04. Every host runs an instance of the prototype implementation and uses one of three hardcoded example policies. All hosts compute a network-wide composite policy using the \wedge_M -operation.

In detail, the runtime of the following three operations was measured: (1) evaluating a local policy, (2) evaluating a composite policy and (3) adding a new host and its local policy to an already existing network of hosts with an already established composite policy. The runtime of the first two operations was measured for three different requests one hundred times each. While the requests 1 and 2 can be evaluated and properly answered by the local policy, the third requests access to an object that is external to the local policy, i. e. the policy cannot make a meaningful decision and directly responds to the request with abstention. For all composite policies which were used for the measurements, a decision could be made for all requests.

The average time it takes to evaluate a local policy is displayed in Fig. 2. In Fig. 3, the average time it takes to evaluate any of the three requests in a tier 1, tier 2 and tier 3 composite policy with a varying number of involved component policies is shown. Lastly, Fig. 4 shows the average time it takes for a host to join an already established network of hosts. This group-formation phase starts when the new host broadcasts its initial message to the network and ends when all hosts, including the arriving one, finished computing the new, network-wide composite policy.

As can be seen, evaluating a local policy for a request that the policy is not concerned with takes significantly less time than evaluating a request the policy can make a decision on.

For the latter, evaluation takes an average of 0.95 ms. In contrast to that, evaluating a composite policy takes more time than evaluating a local policy. As can be seen, the necessary time increases with the number of involved component policies, which is to be expected, as with more component policies, more expressions have to be evaluated in order to compute a result. However, the tier according to which the component policies are composed significantly impacts

⁷The source code is available at <https://codeberg.org/rgorges/acca>.

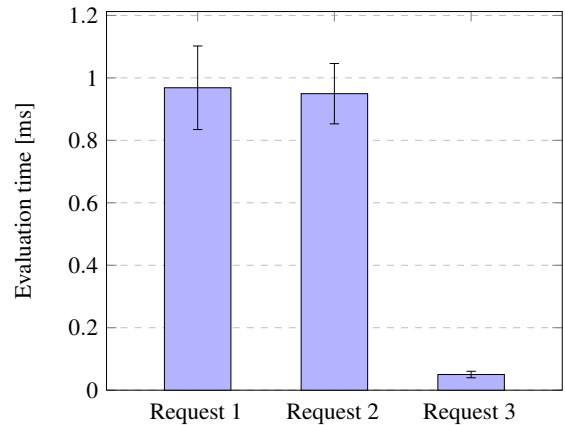


Figure 2: The evaluation times for a local policy. The error bars indicate the standard deviation.

the time needed to evaluate the resulting composite policy. In the prototype implementation, evaluating a tier 3 composite policy is faster than evaluating a tier 1 composite policy, while tier 2 takes the most amount of time. This matches the amount of communication necessary when evaluating the composite policies: In tier 1, exactly one query has to be sent for every component policy which is part of the composite policy. This query asks the host this component policy comes from for the result of its local policy. For tier 2, multiple queries have to be sent for every component policy, namely one per attribution that has to be evaluated. Finally, in tier 3, no queries have to be sent, as all information necessary to evaluate the request in the composite policy are available at the queried host. This results in the described relation of the runtime for the three tiers.

Finally, it can be seen that the effort of incorporating a new host into an already existing network is approximately linear in the number of involved hosts. This is to be expected, as every host in the network adds only a constant number of messages which have to be sent in order to enable the proper incorporation of new hosts. Additionally, this process takes roughly the same time for all three composition tiers, which is because the time necessary for communication between hosts outweighs the effort for computing a composite policy.

Note that these results suggest the use of tier 3 whenever possible, as it comes with the least runtime for evaluating the composite policy, while the effort of computing such a composition is comparable to tiers 1 and 2. However, this is only true as long as the component policies are static. With changing component policies, it becomes necessary to update their copies which were distributed among the other hosts within the network, causing more communica-

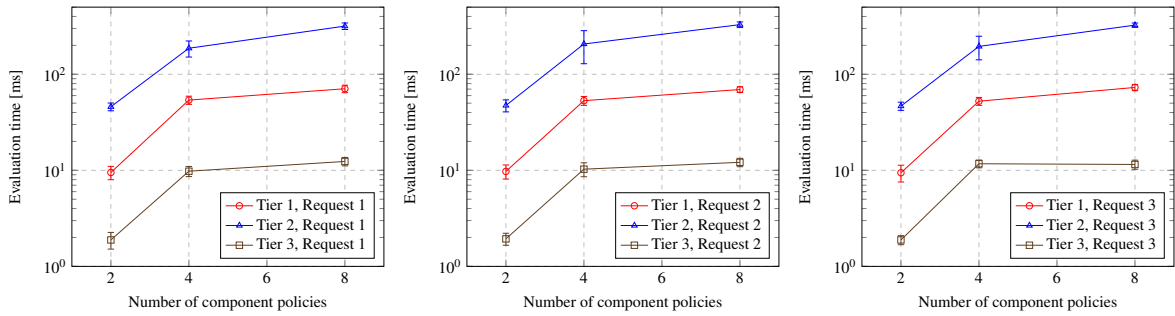


Figure 3: Time to evaluate the three requests in a tier 1, 2 and 3 composite policy with a varying number of component policies as well as for all three of the aforementioned requests. The error bars indicate the standard deviation.

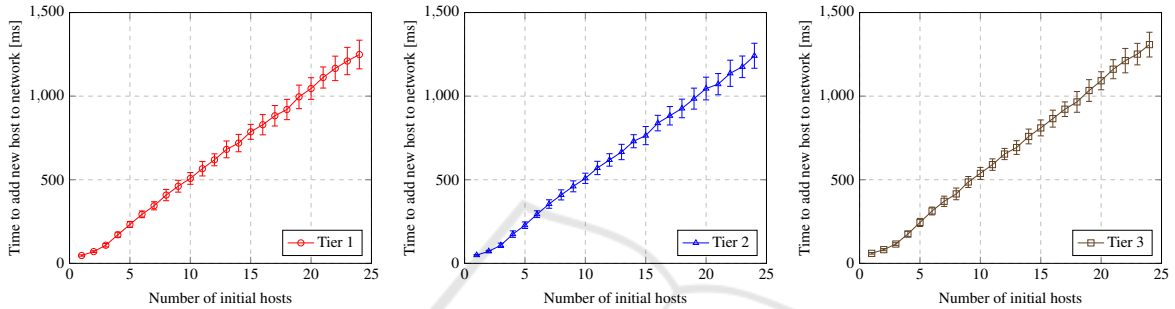


Figure 4: Time to add a host to an already existing network of hosts using the three different composition tiers. This includes computing the new composite policy on every host as well as the communication time. The error bars indicate the standard deviation.⁸

tion overhead, especially since policies may change very frequently because they can incorporate e. g. regularly updated sensor values. Consequently, it is necessary to design and implement such an update mechanism and evaluate its runtime impact to more properly investigate the suitability of the tier 3 policy composition.

8 RELATED WORK

Literature related to this work can be divided in three categories: Policy composition logic in general, specific AC models tailored to distributed V2X systems, and general work on access control technologies in vehicular networks not directly related to policy composition.

Policy Composition. Since a few decades, the policy composition problem and the general semantics of “metapolicies” has received considerable attention (Kühnhauser, 1999; Bonatti et al., 2002; Wijesekera and Jajodia, 2003; Ni et al., 2009; Li et al., 2009;

Ferraiolo et al., 2011; Baracaldo et al., 2011). We designed ACCA on basis of this foundational body of work, but tailored towards the requirements of more recent use cases: first, all these approaches predominantly focus on formally well-established role-based access control (RBAC) policy semantics, which is too restrictive when it comes to modeling object attributes (e. g. in the scenarios described in § 2). Second, and even more important, none of them addresses both observations (2) and (3) as stated in § 1: incomplete knowledge of component policies cannot be tolerated by the policy composition logic, and/or dynamic correctness properties cannot be modeled (and therefore analyzed).

V2X Policy Models. Despite more application-tailored than DABAC, two AC models from the literature are directly related to the approach followed in this work. In (Alsarra et al., 2019), a RBAC model well-tailored to VANETs is described (“openRBAC”). The authors cover its formal definition, inherent role semantics, and hint at its date model implementation and usage. We acknowledge this work as complementary, since openRBAC – being a specialization of DRBAC (Schlegel and Amthor, 2020) which, in turn, is generalized by DABAC – is expected to be completely coverable by ACCA. Substan-

⁸For reasons of hardware availability, these measurements were conducted on an Intel system equivalent to the one described in § 7.

tiating this claim by further incorporation of openRBAC policies in a practical evaluation is subject to future work (cf. §9). In (Gupta et al., 2019), CV-ABAC_G is proposed. In contrast to DABAC, this model distinguishes between more specialized entity classes (such as “clustering” for entity composition, which we represent through domains in the separate system model). On the level on attribution and access rules semantics however, CV-ABAC_G largely matches the intentions of DABAC except for hierarchical attribution and the ability to model environmental attributions (DABAC’s “ext” components). It is therefore a valid alternative to adapt ACCA to more specialized V2X scenarios.

General V2X AC Technology. Finally, significant work has investigated the AC semantics involved with V2X scenarios and VANETs, such as (Sharma and Kaul, 2021; Li et al., 2020; Liu et al., 2022). We regard this literature complementary in a sense that its results will be of relevance for a practical deployment and real-world evaluation of ACCA, as envisioned in §9.

9 CONCLUSION

In this work we present the formal foundations for access control policy composition in a open distributed system, characterized by decentralized enforcement and volatile group formation. As a basis of our approach, we leverage the automaton-based dynamic analysis calculus embedded in the DABAC modeling framework, which serves as a policy specification basis for ACCA, a novel composition algebra for access control policies.

We describe the policy composition problem for V2X applications and apply ACCA for these, which is based on the idea of partial knowledge: for each composition operation, it includes specialized decision making semantics for a group’s composite policy based on one of three *tiers* to represent the degree of information received about multiple DABAC-modeled component policies.

In a first, simulation-based study on runtime performance, we evaluated the utility of these tiers w. r. t. practical feasibility. While our first results suggest an always-flooding-approach for policy information distribution (tier 3), a broader and more application-specific range of dynamic behavior might reveal the utility of tier 1 or even tier 2 policies as a compromise in more detailed use cases. Future work will focus on the investigation of such side conditions and constraints, as well as on a practical infrastructure archi-

ture as a testbed for evaluating a realistic deployment.

AVAILABILITY

A Rust-based implementation of ACCA and the simulation environment used is available at <https://codeberg.org/rgorges/acca>.

REFERENCES

- Alsarra, S., Yen, I.-L., Huang, Y., Bastani, F., and Thuraisingham, B. (2019). An OpenRBAC semantic model for access control in vehicular networks. In *Proceedings of the 24th ACM Symposium on Access Control Models and Technologies*, SACMAT ’19, page 93–102, New York, NY, USA. Association for Computing Machinery.
- Anderson, J. P. (1972). Computer Security Technology Planning Study. Technical Report ESD-TR-73-51, Air Force Electronic Systems Division, Hanscom AFB, Bedford, MA, USA. Also available as Vol. I, DITCAD-758206. Vol. II DITCAD-772806.
- Arena, F. and Pau, G. (2019). An overview of vehicular communications. *Future Internet*, 11(2):27.
- Ashutosh, A., Gerl, A., Wagner, S., Brunie, L., and Kosch, H. (2023). XACML for mobility (XACML4M)—an access control framework for connected vehicles. *Sensors*, 23(4):1763.
- Baracaldo, N., Masoumzadeh, A., and Joshi, J. (2011). A Secure, Constraint-Aware Role-Based Access Control Interoperation Framework. In *Proceedings of the 5th International Conference on Network and System Security*, NSS ’11, pages 200–207. IEEE Computer Society.
- Bonatti, P., De Capitani di Vimercati, S., and Samarati, P. (2002). An Algebra For Composing Access Control Policies. *ACM Transactions on Information and System Security*, 5:1–35.
- Ferraiolo, D., Atluri, V., and Gavrila, S. (2011). The Policy Machine: A Novel Architecture and Framework for Access Control Policy Specification and Enforcement. *Journal of Systems Architecture: the EUROMICRO Journal*, 57(4):412–424.
- Ferraiolo, D., Chandramouli, R., Kuhn, R., and Hu, V. (2016). Extensible access control markup language (xacml) and next generation access control (ngac). In *Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control*, ABAC ’16, pages 13–24, New York, NY, USA. ACM.
- Gupta, M., Benson, J., Patwa, F., and Sandhu, R. (2019). Dynamic groups and attribute-based access control for next-generation smart cars. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*, CODASPY ’19, page 61–72, New York, NY, USA. Association for Computing Machinery.

- Harrison, M. A., Ruzzo, W. L., and Ullman, J. D. (1975). On Protection in Operating Systems. *Operating Systems Review, special issue for the 5th Symposium on Operating Systems Principles*, 9(5):14–24.
- Hussein, N. H., Yaw, C. T., Koh, S. P., Tiong, S. K., and Chong, K. H. (2022). A comprehensive survey on vehicular networking: Communications, applications, challenges, and upcoming research directions. *IEEE Access*, 10:86127–86180.
- INCITS (2013). Information technology — Next Generation Access Control — Functional Architecture (NGAC-FA). INCITS 499–2013, American National Standards Institute.
- Khan, A. and Fong, P. (2012). Satisfiability and Feasibility in a Relationship-Based Workflow Authorization Model. In Foresti, S., Yung, M., and Martinelli, F., editors, *Computer Security – ESORICS 2012*, volume 7459 of *Lecture Notes in Computer Science*, pages 109–126. Springer Berlin / Heidelberg.
- Kühnhauser, W. E. (1999). Policy Groups. *Elsevier Computers & Security*, 18(4):351–363.
- Li, H., Pei, L., Liao, D., Chen, S., Zhang, M., and Xu, D. (2020). FADB: A fine-grained access control scheme for VANET data based on blockchain. *IEEE Access*, 8:85190–85203.
- Li, N., Mitchell, J. C., and Winsborough, W. H. (2005). Beyond Proof-of-compliance: Security Analysis in Trust Management. *Journal of the ACM*, 52(3):474–514.
- Li, N., Wang, Q., Qardaji, W., Bertino, E., Rao, P., Lobo, J., and Lin, D. (2009). Access Control Policy Combining: Theory Meets Practice. In *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies*, SACMAT '09, pages 135–144, New York, NY, USA. ACM.
- Liu, L., Chen, C., Pei, Q., Maharjan, S., and Zhang, Y. (2020). Vehicular edge computing and networking: A survey. *Mobile Networks and Applications*, 26(3):1145–1168.
- Liu, X., Chen, W., and Xia, Y. (2022). Security-aware information dissemination with fine-grained access control in cooperative multi-RSU of VANETs. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):2170–2179.
- Marinovic, S., Craven, R., Ma, J., and Dulay, N. (2011). Rumpole: a Flexible Break-glass Access Control Model. In *Proceedings of the 16th ACM Symposium on Access Control Models and Technologies*, SACMAT '11, pages 73–82. ACM.
- Mazzola, M., Schaaf, G., Niewels, F., and Kurner, T. (2015). Exploration of centralized Car2X-systems over LTE. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*. IEEE.
- Meneguette, R., Grande, R. D., Ueyama, J., Filho, G. P. R., and Madeira, E. (2021). Vehicular edge computing: Architecture, resource management, security, and challenges. *ACM Computing Surveys*, 55(1):1–46.
- Ni, Q., Bertino, E., and Lobo, J. (2009). D-algebra for Composing Access Control Policy Decisions. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ASIACCS '09, pages 298–309, New York, NY, USA. ACM.
- OASIS (2013). eXtensible Access Control Markup Language (XACML) Version 3.0. OASIS Standard 499–2013, Organization for the Advancement of Structured Information Standards.
- Sandhu, R. S. (1992). The Typed Access Matrix Model. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy*, SP '92, pages 122–136, Washington, DC, USA. IEEE Computer Society.
- Schlegel, M. and Amthor, P. (2020). Beyond Administration: A Modeling Scheme Supporting the Dynamic Analysis of Role-based Access Control Policies. In Samarati, P., di Vimercati, S. D. C., Obaidat, M. S., and Ben-Othman, J., editors, *Proceedings of the 17th International Conference on Security and Cryptography*, SECURITY 2020, pages 431–442. INSTICC, SciTePress.
- Schlegel, M. and Amthor, P. (2021). The Missing Piece of the ABAC Puzzle: A Modeling Scheme for Dynamic Analysis. In di Vimercati, S. D. C. and Samarati, P., editors, *Proceedings of the 18th International Conference on Security and Cryptography*, SECURITY 2021, pages 234–246. INSTICC, SciTePress.
- Schlegel, M. and Amthor, P. (2023). Putting the Pieces Together: Model-Based Engineering Workflows for Attribute-Based Access Control Policies. In Samarati, P., van Sinderen, M., di Vimercati, S. D. C., and Wijnhoven, F., editors, *E-Business and Telecommunications*, volume 1795 of *Communications in Computer and Information Science (CCIS)*, pages 249–280. Springer Nature Switzerland, Cham.
- Sharma, S. and Kaul, A. (2021). VANETs cloud: Architecture, applications, challenges, and issues. *Archives of Computational Methods in Engineering*, 28(4):2081–2102.
- Wijesekera, D. and Jajodia, S. (2003). A propositional policy algebra for access control. *ACM Trans. Inf. Syst. Secur.*, 6(2):286–325.
- Yang, C., Jiang, P., and Zhu, L. (2022). Accelerating decentralized and partial-privacy data access for VANET via online/offline functional encryption. *IEEE Transactions on Vehicular Technology*, 71(8):8944–8954.
- Zhou, Z., Gaurav, A., Gupta, B. B., Lytras, M. D., and Razak, I. (2022). A fine-grained access control and security approach for intelligent vehicular transport in 6G communication system. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):9726–9735.