Classifying and Comparing Attribute-Based and Relationship-Based Access Control

Tahmina Ahmed
Univ. of Texas at San Antonio
qfk367@my.utsa.edu

Ravi Sandhu
Univ. of Texas at San Antonio
ravi.sandhu@utsa.edu

Jaehong Park
Univ. of Alabama in Huntsville
jae.park@uah.edu

ABSTRACT

Attribute-based access control (ABAC) expresses authorization policy via attributes while relationship-based access control (ReBAC) does so via relationships. While ABAC concepts have been around for a long time, ReBAC is relatively recent emerging with its essential application in online social networks. Even as ABAC and ReBAC continue to evolve, there are conflicting claims in the literature regarding their comparison. It has been argued that ABAC can subsume ReBAC since attributes can encode relationships. Conversely there are claims that the multilevel (or indirect) relations of ReBAC bring fundamentally new capabilities. So far there is no rigorous comparative study of ABAC vis a vis ReBAC.

This paper presents a comparative analysis of ABAC and ReBAC, and shows how various ReBAC features can be realized with different types of ABAC. We first identify several attribute types such as entity/non-entity and structured attributes that significantly influence ABAC or ReBAC expressiveness. We then develop a family of ReBAC models and a separate family of ABAC models based on the identified attribute types, with the goal of comparing the expressive power of these two model families. Further, we identify different dynamics of the models that are crucial for model comparison. We also consider different solutions for representing multilevel relationships with attributes. Finally, the ABAC and ReBAC model families are compared in terms of relative expressiveness and performance implications.

Keywords

Access Control; ABAC; ReBAC; Attribute; Relationship

1. INTRODUCTION & MOTIVATION

The concept of using attributes for access control has been around for many years, e.g., the X.500 standard [16] was an early effort for managing object information with attributes. Attribute-based access control (ABAC) is considered one of the most generalized forms of access control as it can capture the salient features of discretionary access control (DAC), mandatory access control (MAC) and role-based access control (RBAC) using appropriate attributes such as access control lists, security labels and roles respectively [35], and bring in additional elements such as location and time. ABAC enables more precise access control as it can consider a higher number of discrete inputs into an access control decision [33]. Different ABAC models with rich policy languages and sophisticated features have been proposed [34, 35, 36, 40, 48, 51].

Meanwhile, in recent years, various online social network (OSN) applications such as Facebook, Twitter and LinkedIn have become widely used. In OSNs, authorization for users' access to specific content is typically based on the interpersonal relationships between the accessing user and content owner. OSN ReBAC models mostly use user-to-user relationships [12, 15, 21, 22, 29, 30] while user-to-resource and resource-to-resource relationships have also been considered in some cases [13, 20]. Several access control models have been proposed for OSN ReBAC considering both single and multiple relationship types for authorization policy specification [13, 20, 21, 29]. Subsequently, additional models have been proposed to extend and generalize these OSN ReBAC models so that they can be applicable to computing systems beyond OSNs [7, 24, 28, 43].

ABAC has been around for a long time and can be viewed as a generalization, unification and extension of earlier access control concepts including discretionary, mandatory and role-based access control. ReBAC is relatively recent, with its initial motivation stemming from its essential application in online social networks but now generally regarded as having broader applicability. Both have considerable applications in industry, and are anticipated to continue being important for the foreseeable future.

From an ABAC perspective it is often claimed that attributes can express relationships [51], and indeed this is trivial for direct relationships such as a friend relation between two users [24]. However, the use of indirect relations, also called multilevel or composite relations, is fundamental to ReBAC [12, 24]; a familiar example being friend of friend. It is hard to see how ABAC can express long chains of relationships [24]. It has been suggested that ReBAC emerged to overcome this shortcoming of attributes [12].

Any attempt to compare ABAC and ReBAC is made additionally difficult by the fact that there are no consensus models for either one that are widely accepted. Rather both arenas exhibit a proliferation of models which are continuing to evolve as different aspects of each arena are explored.

CODASPY'17, March 22-24, 2017, Scottsdale, AZ, USA

DOI: http://dx.doi.org/10.1145/3029806.3029828

In this paper we develop a rigorous comparison between ABAC and ReBAC. We do this by classifying ABAC and ReBAC models based on salient aspects that are relevant to their comparison. The main purpose of these classifications is to enable comparison. The classifications are not intended to be a complete characterization of ABAC models or ReBAC models. They are only a partial classification but sufficient to draw out the essential relationships between ABAC and ReBAC.

The rest of the paper is organized as follows. Section 2 provides appropriate background on ABAC and ReBAC for our purpose. Section 3 presents attribute types, characteristics, definitions and some assumptions on attributes. Sections 4 and 5 provide the classification of ReBAC and ABAC models respectively with structural variations and dynamics, giving us a family of ReBAC models and a separate family of ABAC models. Section 6 identifies two solutions for expressing multilevel relations with attributes. Section 7 compares the models in the ABAC and ReBAC families. Section 8 concludes the paper.

2. BACKGROUND

This section provides an overview of ABAC and ReBAC models, relevant to our purpose.

2.1 ReBAC Models

As OSNs have gained popularity, several ReBAC models have been introduced to capture various authorization policies. More recently, researchers have proposed extended ReBAC models applicable to other computing systems beyond OSNs. In this subsection, we review these two types of ReBAC models.

2.1.1 ReBAC for Online Social Networks

Fong et al. [29] presented a Facebook-like access control model, featuring four types of policies that cover four different aspects of access in OSNs. The four policies include user search, traversal of the social graph, communication between users and normal access to objects owned by users. The policy vocabulary supports expressing some topology-based properties, such as k common friends and k clique. The model uses single relationship types between users.

Carminati et al. [15] proposed an access control model which considers type, depth and trust metrics of user-to-user relationship between accessing user and target user. It also considers multiple types of relationships between users. In [13], Carminati et al. proposed a model which utilizes semantic web technology. This model considers multiple type relationships between users and resources.

Cheng et al. [21] proposed a user-to-user relationship based access control model with a regular expression-based policy specification language. Social graph of UURAC contains user-to-user relationships only. The connection between resources and users are referred to as controlling user (e.g., owner, tagged user). URRAC model [20] extends UURAC to include user-to-resource and resource-to-resource relations. In both models, multiple types of relationships are supported, and policy language can specify relationship path patterns between accessing user and target resource or user.

Subsequently Cheng et al. [22] defined an attribute-aware ReBAC model to express the contextual nature of relationships and users. The authors have extended their UURAC model to incorporate node attributes and relationship attributes. They further introduced the concept of a graph attribute such as count which is associated with the relationship graph other than with a particular node or edge.

Bennett et al. [9] proposed a ReBAC model that considers multiple types of relationships between users and demonstrates how conflicts and potential misconfigurations can be automatically detected using the Alloy Analyzer [1]. Pang et al. [39] proposed an access control scheme for OSN where they have taken hybrid logic approach to use public information along with relationships.

2.1.2 ReBAC Beyond Online Social Environment

Fong et al. [28] proposed a formal ReBAC model intending to widen the application of ReBAC beyond social computing. The model considers multiple relationship types between users with directional relationships and access contexts, and uses a modal logic language for policy specification. The connection between users and resources is maintained through a system function called 'resource owner.' Fong et al. [30] extended the policy language of [28] and characterized its expressiveness. Subsequently they defined hybrid logic for ReBAC which can express complex relationship requirements [12].

Crampton et al. [24] proposed the RPPM model that can be applied to general computing system. The model considers users, resources and other logical and physical entities (i.e., files, folders, organizations, etc.) as nodes of a labeled relationship graph. Policies are defined using path conditions. The model allows multiple types of relationship between different entities. The model uses a two-stage decision process: it first computes the path between requester and the requested resource and tries to find matches from a list of policies, and then it determines whether those policies are authorized. Rizvi et al. [43] demonstrated an implementation of RPPM model in an open-source medical record system. Subsequently they extended their model to be interoperable with RBAC [42]. Recently Crampton et al. [26] proposed a framework for inter-operating multiple ReBAC model instances by initiating request in one system to target resource in a second system.

Most ReBAC models consider user-to-user and possibly user-to-resource relationships. Very few of consider resource-to-resource relationships. Models that consider resource-to-resource relationships typically do so through users. Recently Ahmed et al. [7] proposed a ReBAC model which considers object-to-object relationships without intervening users, and demonstrated an implementation of the model in the OpenStack's [3] object storage, Swift [6].

All the models reviewed so far are operational models. Recently a number of ReBAC administrative models have also been proposed for general purpose ReBAC [19, 25, 49] which consider graph dynamics such as adding/deleting nodes (entities) and or edges (relationships). In particular, [19] introduces the concept of dependent edge in ReBAC and considering dependencies during edge deletion.

2.2 ABAC Models

ABAC has been studied for a long time and many different formal models have been proposed [34, 35, 36, 40, 48, 51]. Several of these are application specific or limited to a specific domain. ABAC for web services [51] proposed an ABAC model for web service authorization, while [48] defined an ABAC model for semantic web technology. UCON [40]

was proposed to capture authorization continuity and attribute mutability. [36] defines an ABAC model for service oriented architecture considering requester's privacy preference. $ABAC_\alpha$ [35] is proposed to configure DAC, MAC and RBAC, while $ABAC_\beta$ [34] extends $ABAC_\alpha$ to incorporate different RBAC extensions. NIST ABAC [33] provides a detail explanation of ABAC concepts and considerations for deployment of enterprise ABAC capabilities. XACML [2] proposes a standardized mechanism to specify ABAC authorization policy, request and policy evaluation. Attribute-based encryption is supports fine-grained sharing of encrypted data [11, 17, 37, 38, 41, 44].

3. ATTRIBUTES

In our comparison and classification for ReBAC and ABAC models, attributes play an important role. In this section we identify and discuss various types of attributes based on several different criteria. Some of these attribute types are crucial for ABAC and ReBAC comparison as their existence in a model strongly influences its expressiveness and performance. Others are not quite significant for our comparison purpose. In the next two sections, we use these attribute types to classify ReBAC and ABAC models to facilitate comparison between them.

3.1 Attribute Types

In this subsection we present several attribute types classified using five different criteria. Specifically the criteria are based on (1) how attribute value(s) are structured, (2) what the attribute scope is, (3) boundedness of attribute range, (4) attribute association and (5) attribute mutability.

Depending upon the type of attribute value, there can be three types of attributes.

- **Atomic-valued or Single-valued Attribute:** If an attribute has at most one value associated with it at any one point in time, it is called atomic-valued or single-valued [4, 35] attribute. For example, gender attribute can have only a single value at a given time.
- **Set-valued or Multi-valued Attribute:** If an attribute can have more than one value associated with it at any one point in time, it is called set-valued or multi-valued attribute. For example, a person can have more than one phone number [4, 35].
- **Structured Attribute**: A structured attribute has a number of single or multi-valued sub-attributes [5]. For example, a Person-Info attribute can have sub-attributes of name, age and phoneNumber.

Depending upon the scope of the attribute, attributes can be either Entity Attribute or Non-entity Attribute.

- **Entity Attribute:** An entity is a thing which can be distinctly identified. A specific person, company an object or event is an example of entity [18]. Entity attribute takes an entity as input and returns another entity, a set of entities, or a structured tuple containing at least one entity. For example, an attribute value of parent of a person, owner of an object or friend of a person is another person (entity).
- **Non-entity Attribute:** Attributes whose range is not defined on the set of entities in the system are called non-entity attributes. For example, user's age or gender does not include another entity as its value. The concept of non-entity attribute depends upon what is defined as entities in the system. For example, suppose roles or organizations are entities in a system, and the range of attributes "assigned-roles" and "worksAt" are a set of roles and a set of organizations, respectively. In that case both attributes are entity attributes. If roles and organizations are not defined as entities in the system, these are non-entity attributes.

Depending upon whether the range of an attribute is bounded or not, attributes can be either finite domain attribute or infinite domain attribute.

- **Finite Domain Attribute:** Range of this attribute type is a finite set of attribute value (e.g., gender, role).
- **Infinite Domain Attribute:** Range of this attribute type is a countably infinite set of attribute values (e.g, time). Entity attributes where new entities can be created without bound are infinite domain attributes.

Considering the association of an attribute we can have two types of attributes [33, 34, 51]

- **Contextual or Environmental Attribute**: These attributes are independent and not associated with any specific users, subjects, objects or entities in the system. They are global and managed by the system and associated with system. For example, *current-time* is system-wide information and not associated with a specific entity [34]. Other examples include system status, network security level, and so on [33, 51].
- **Meta Attribute**: Meta attributes are attributes of an attribute. Unlike regular attributes that are associated with entities, meta attributes are associated with other attributes. For example a user is associated with a role and the role is associated with a task. Here, the role is an attribute, and the task is a meta attribute [34].

Considering the mutability of attributes there are two types of attribute [40].

- **Mutable Attribute:** Mutable attributes are changed as a consequence or side effect of users' access or activity.
- **Immutable Attribute:** Immutable attributes can be changed only by direct administrative activity of a user or administrator.

The notions of entity/non-entity, finite/infinite domain, atomic-valued/set-valued/structured attributes are important for ReBAC-ABAC comparison as they are key attribute types that will strongly influence expressibility of relationships between entities or configurability of relationship graph.

Unlike these key attribute types, contextual/environmental attribute is a special type of attribute, not related to entities. Meta attribute defines relationship between attributes. Mutability is special feature specified in usage control for consumable authorization. These type of attributes are not relevant to ReBAC-ABAC comparisons with respect to expressiveness or performance. In the next two subsections, we will further discuss the definitions of these key attribute types and some assumptions for the rest of the paper.

3.2 Attribute Definitions for ReBAC and ABAC Comparison

For our ReBAC and ABAC comparison, we consider entity and non-entity, finite and infinite domain, atomic-valued, set-valued and structured attributes. In this subsection, we define these key attribute types (except for single-valued, multi-valued and structured attributes which have been adequately defined above).

DEFINITION 1. *Entity Attribute: An attribute att_i is an entity attribute if*

i. range of att_i is a set of entities (i.e. att_i: $E_j \rightarrow E_k$),

ii. range of att_i is a powerset of entities (i.e. att_i: $E_j \rightarrow 2^{E_k}$), or

iii. att_i is a structured attribute with at least one sub-attribute being an entity attribute.

For example, if user is defined as an entity in the system and best-friend is an atomic or set-valued attribute on user then best-friend is an entity attribute. At a specific time each entity set is fixed but can change over time if the system allows entity changes (i.e., creation or deletion of entities.). If att_i is a structured attribute and at least one sub-attribute of att_i is an entity attribute then att_i is also an entity attribute. For example let's say 'roleInfo(roles,assignedby)' is a structured attribute which has 'roles' and 'assignedby' as sub-attributes. Here 'roles' is non-entity attribute whose range is set of roles however 'assignedby' is an entity attribute whose range is set of users. So 'roleInfo' is an entity attribute.

DEFINITION 2. *Non-Entity Attribute: An attribute att_i is a non-entity attribute if it is not an entity attribute.*

Examples are phoneNumber and age. Note that if att_i is a structured attribute then every sub-attribute of att_i must be a non-entity attribute for att_i to be a non-entity attribute.

DEFINITION 3. *Finite Domain Attribute: An attribute domain is finite if the range of the attribute doesn't grow over time.*

For example, 'gender' is a finite domain attribute. Also, 'roles' and 'security clearance' are finite domain attributes if the system does not allow new roles or security clearances to be added over time.

DEFINITION 4. *Infinite Domain Attribute: An attribute domain is infinite if the range of the attribute grows over time.*

For example, in an OSN, if a new user can be created so he or she can be a friend of other users, the friend attribute is an infinite domain attribute as the range of friend is changed over time.

Finally, we introduce the familiar concept of attribute function composition [8, 31].

DEFINITION 5. *Attribute Function Composition: Nesting two or more attribute functions to form a single new function is known as attribute function composition. The composition of two attribute functions $f: X \rightarrow Y$ and $g: Y \rightarrow Z$ yields a function which maps $x \in X$ to $g(f(x)) \in Z$. Composition is denoted as $g \circ f$, where g is a function whose domain includes the range (or codomain) of f. We write $(g \circ f)(x)=g(f(x))$.*

A function $h(x) = f_n(.....f_2(f_1(x))....)$ which is the composition of n functions (same or different), say f_1 to f_n, is also said to be a composite function. Intuitively, composing two or more functions is a chaining process in which the output of the first function becomes the input of the second one, and the output of the $(k\text{-}1)^{th}$ function becomes the input of the k^{th} function.

3.3 Assumptions

For ease of our comparison, all the ReBAC and ABAC models considered in this paper comply with the following assumptions.

1. *All non-entity attributes are finite domain.* Attributes such as role, department, title, gender, etc., typically admit only a small number of finite values by their intrinsic nature. Attributes such as location can be ever finer grained, so in principle could be regarded as infinite domain but a large finite domain should be adequate. Time being modeled as a finite domain has similar issue. For our purpose a finite domain assumption is reasonable.

2. *Each entity has a countably infinite set for all possible entities of that type.* For example if users, subjects and objects are the only entities defined in a particular system then the countably infinite sets for users, subjects and objects are $\mathcal{U}$, $\mathcal{S}$ and $\mathcal{O}$. The existing set of users, subjects and objects at any moment are U, S, O respectively where U, S, O are finite sets, and $U \subset \mathcal{U}$, $S \subset \mathcal{S}$ and $O \subset \mathcal{O}$.

3. *Identity of an entity is not reusable.* If an entity gets deleted, its identity cannot be used for another entity that is created after the deletion.

4. *All entity attribute functions are partial functions defined on existing entities only.* For example let $\mathcal{U}$ is the countably infinite set of all possible users, and U the finite set of current users ($U \subset \mathcal{U}$). An entity attribute function $f: U \rightarrow Y$ is defined only for elements of U and is undefined for elements in $\mathcal{U}$-U. We understand $f: U \rightarrow Y$ for an entity set U to mean that U will change with time but is finite at any moment. Note that if the system allows creation of entities then the entity attributes have infinite or unbounded domain. If the system doesn't allow any entity creation or deletion then the entity attributes form a finite domain.

5. *For attribute function composition inner attribute functions should always be entity attributes.* We require that a non-entity attribute can only occur as the outermost function in a composition. So for a composition $f_n(.....f_2\ (f_1(x))....)$, for $1 \leq i \leq n\text{-}1$, f_i must be an entity attribute function, while f_n can be either entity or non-entity attribute.

6. For any set valued attribute function f defined on set X, we understand $f(X) = \bigcup_{x_i \in X} f(x_i)$. So an attribute function composition friend(friend("Alice")) means: $\bigcup_{u_i \in friend(\text{“}Alice\text{”})} friend(u_i)$

7. We understand that structured attribute is a multivalued tuple of atomic and or set-valued attributes. So it is more expressive than atomic or set valued attributes.

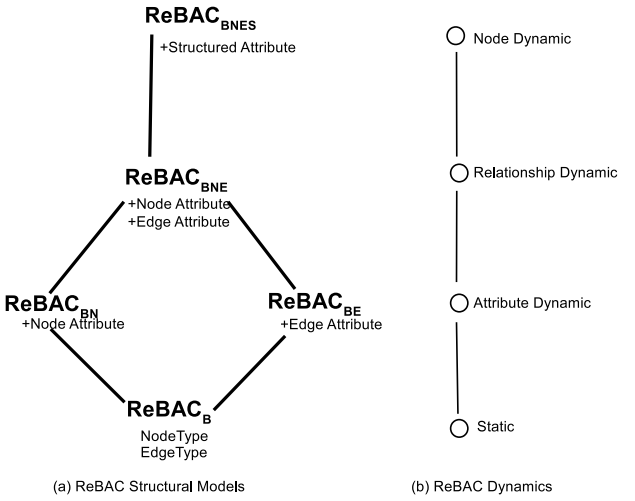

Figure 1: *ReBAC Framework*

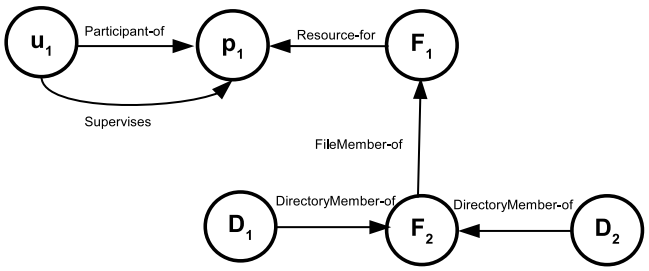

Figure 2: *An Example of a Relationship Graph Expressible in $ReBAC_B$ [24]*

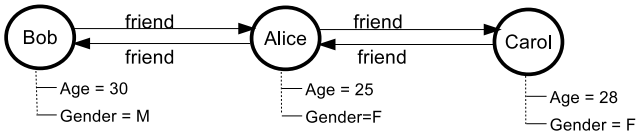

Figure 3: *An Example of Node Attributes in Relationship Graph Expressible in $ReBAC_{BN}$*

Structured attribute can express atomic or set-valued attribute by having a single sub-attribute.

4. ReBAC CLASSIFICATION

In this section we develop a ReBAC framework including a family of structural models. The framework is illustrated in Figure 1 and consists of two components. Specifically, Figure 1(a) shows a family of structural models while Figure 1(b) shows the different types of dynamics found in ReBAC models.

The goal of this framework is to build a classification of ReBAC models that facilitates comparison with ABAC models. While there are many sophisticated proposals on ReBAC policy expression mechanisms such as incoming versus outgoing policy, policy individualization, modal/hybrid/ first order/propositional logic based policies, this framework does not focus on policy specification. Rather it is independent of policy languages and focusses on structural and dynamic aspects of ReBAC.

Figure 1(a) depicts ReBAC models with increasing capabilities as we go upwards in this hierarchy. In ReBAC, entities are represented as nodes in a relationship graph, and relations as entity to entity edges. We use the terms "node" and "entity" as synonyms, and likewise for the terms "edge" and "relation". The base model ReBAC_B allows for multiple node types (e.g., user, resource project, organization, group, etc.) and multiple directed or undirected edge types (e.g., friend, coworker, spouse, parent, etc.) Figure 2 shows an example relationship graph [24] expressible in ReBAC_B. Most of the relationship graphs permitted in existing ReBAC models, including [20, 21, 24, 28, 29], can be expresed with the capabilities of ReBAC_B.

ReBAC_BN adds node attributes to ReBAC_B. Node attributes enable consideration of entity attributes along with relationships in authorization policies. For example, in a professional social network we may have a policy that an employee of an organization o_1 can connect to a recruiter of organization o_2 only if the recruiter is not already connected to any employees of o_1. In this case, the organization attribute of users (nodes) needs to be considered along with professional relationships. Another example is an online dating site where a single male user wants to connect a single female who has less than 4^{th} degree connection with him through only his female friends and is at least two years younger than him. Here we need to consider gender, age and relationship depth along with relationships. Such attribute-aware ReBAC is discussed in greater detail in [22]. Figure 3 shows an example relationship graph with node attributes.

ReBAC_BE extends ReBAC_B with edge attributes. For example, some ReBAC models use trust value of relationships to show the connection strength between users [14, 15]. In general, when a ReBAC authorization policy needs to consider some properties of relationships beyond relationship types, the relationship graph needs edge attributes to store and express those criteria, such as proposed in [22]. Figure 4 provides an example of edge attributes in relationship graph. Here "Bob" is assigned to supervise "Project_1" and "assignedBy" is an edge attribute for relationship type "supervises" which specifies who has assigned "Bob" as supervisor. Similarly "tenant_1" has "tenantTrust" relationship with "tenant_2" and here "trustValue" specifies the strength of how much "tenant_2" trusts "tenant_1".

ReBAC_BNE brings together the two separately motivated extensions of ReBAC_BN and ReBAC_BE, such as in [22]. Following common practice, node and edge attributes in these models are atomic or set-valued attributes.

Recently Cheng et al. [19] proposed a ReBAC administrative model where they introduced the concept of dependent edge in relationship graph. A dependent edge example of MT-RBAC [19] is shown in Figure 5. Here user u owned by tenant x (with relationship type UO) can be "assigned to" a role r (with relation type UA) which is "owned by" tenant y (with relationship type RO) only if tenant y trusts tenant x (with relationship type TT). This particular tenant-trust relationship needs to be considered during role assignment or any time the trust-relationship between x and y changes. If the tenant's trust relationship is revoked at some point of time, the role assignment needs to be revoked as well. In order to configure this scenario using attributes, we need to store a paired set of the role values and the required trust relationship. This additional information allows the

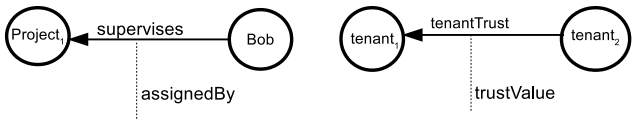

Figure 4: *An Example of Edge Attributes in Relationship Graph Expressible in $ReBAC_{BE}$*

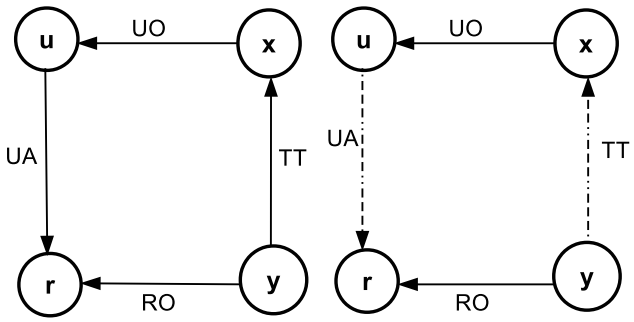

Figure 5: *Example of Dependent Edge Expressible in $ReBAC_{BNES}$ [19]*

model to consider cascading revocation [10, 27, 32] of dependent edges. This edge dependency in a graph cannot be captured using edge types or atomic or set valued edge attributes. To be precise, we will need structured attributes which can store multiple relevant attributes as a single attribute in a certain structure. For the above scenario, the structured attribute can store information of those edges that are required to create another edge. For instance, "dependsOn" attribute of relationship type UA can store a tuple of three sub-attributes: (sourceNode, targetNode, relationshipType), hence, (y,x,TT) for the example above. Consider another example where "securityLabel" is an object attribute. If a graph needs to store the information who has assigned a particular "securityLabel" to an object, we can use a structured attribute where sub-attributes are (label, assignedBy). If relationship graph only considers atomic-/set-valued attributes it won't be able to store this information. Our final model $ReBAC_{BNES}$ considers structured attributes for both nodes and edges. This completes our discussion of Figure 1(a).

Considering the changes or dynamism in ReBAC there are 4 dynamics shown in Figure 1(b). The dynamics are as follows.

- **Static:** In a static ReBAC model, attribute values, nodes and edges of the graph remain unchanged. A static graph is used for access only. Actions such as add or delete relationship between two entities (add or delete edges in the relationship graph), add or delete entities (add or delete nodes in relationship graph are not allowed) and change of attribute values are not allowed.

- **Attribute Dynamic**: ReBAC that allows changes of node attribute and edge attribute values are attribute dynamic ReBAC. For example, consider Hobby is a node attribute of users in a social network. Suppose Hobby("Alice")={gardening, painting}. Recently "Alice" gets interested to do "knitting" and wants to change her hobby in the social network site. If the system allows her to update her hobby as Hobby("Alice") = {gardening, painting, knitting} then it is an attribute dynamic ReBAC.

- **Relationship Dynamic**: ReBAC that allows changes of relationships between entities (add or delete edges in the relationship graph) is called relationship dynamic. Examples include establishing a new relationship between two entities, or deleting an existing relationship between two entities. We consider relationship dynamic includes attribute dynamic for the ReBAC models which have edge attributes, as adding new relationship needs to assign attribute values of that edge.

- **Node Dynamic**: ReBAC that allows changes of entities is called as node dynamic ReBAC. Some examples are creating or deleting a user or resource in a relationship graph. Here we consider node creation implies possible relationship establishment and attribute value assignments when ReBAC models have attributes for nodes and or edges. Hence, node dynamic includes attribute dynamic (for some cases) and relationship dynamic.

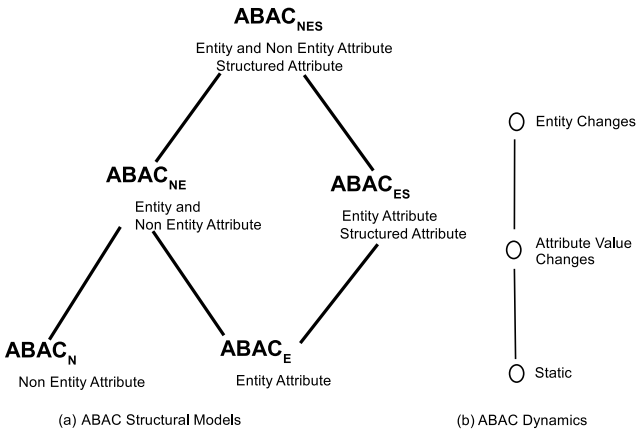

Figure 6: *ABAC Framework*

Each ReBAC dynamic can be combined with any of the ReBAC structural models excluding $ReBAC_B$. $ReBAC_B$ can only have static, relationship dynamic and node dynamic. However $ReBAC_B$ cannot have attribute dynamic as it doesn't have any attributes. Thus, attribute dynamism is irrelevant for $ReBAC_B$.

5. ABAC CLASSIFICATION

In this section, we develop a set of structural models for ABAC with capabilities to configure the ReBAC models defined in Section 4. We define the ABAC models by considering attribute types that are necessary to capture relationships and relationship graphs as shown in Figure 6(a). Specifically, we consider entity/non-entity, finite/infinite domain, and atomic-valued/set-valued/structured attributes. As shown in Figure 6(b), we also identify the dynamics of ABAC models. While this is not the most general framework for ABAC, it facilitates comparative analysis of relative expressiveness of ABAC and ReBAC.

Figure 6(a) depicts ABAC models with increasing capabilities as we go upwards in this hierarchy. $ABAC_N$ considers non-entity attribute only. According to our assumption 5 in Section 3, non-entity attribute cannot configure relationship composition, hence $ABAC_N$ is incomparable to $ReBAC_B$. $ABAC_N$ can only have attributes such as name, gender, location etc.

$ABAC_E$ considers entity attributes only and can configure $ReBAC_B$ model which has multiple relationship types and multiple entity types. Most of the ReBAC models fall under this category [20, 21, 24, 28, 29]. For example, consider the system graph in Figure 2 [24]. To configure it with $ABAC_E$ we need the following.

- entity types = {user, project, file, directory}
- user attributes = {Participant-of, Supervises},
  file attributes ={Resource-for, FileMember-of},
  project attributes = {},
  directory attributes ={DirectoryMember-of}.

$ABAC_{NE}$ considers both entity and non-entity attributes which is similar to considering node attributes along with multiple relationship types and multiple entity types as in $ReBAC_{BN}$. For example, in Figure 2, suppose the user has attributes {name, gender, age} and files have attributes {securityLabel, size}. Using $ABAC_{NE}$, we can configure these node attributes with non-entity attributes.

$ABAC_{ES}$ considers structured entity attributes which can configure relationships and edge attributes of relationship graph. Figure 4 shows some simple edge attributes in relationship graphs. To configure the relationship graph "Bob supervises $Project_1$" in ABAC, we need to have entity attribute "supervises" for user so we can express supervises (Bob)= {"$Project_1$"}. In addition, to express the edge attribute "assignedBy", we will need a structured attribute of user "assignedBy", so we can express assignedBy(Bob) = ("$Project_1$","supervises", "Alice"). Here the sub-attributes for "assignedBy" are (targetNode, relationshipType, assignedByUser). The same is true for the tenantTrust relationship between $tenant_1$ and $tenant_2$. Here we can configure the trustValue with structured attribute trustValue ($tenant_2$) = ("$tenant_1$", 'tenantTrust', 0.5). Consider the example in Figure 5 where the edge (u, r, UA) is dependent on edge (y, x, TT) and this dependency can be represented using a structured attribute for edge. To configure this structured edge attribute in ABAC, we need to have dependentEdge(u) = ("r","UA", {(y,x,TT)}). $ABAC_{ES}$ is comparable to $ReBAC_{BE}$.

$ABAC_{NES}$ considers entity and non-entity structured attributes which can configure relationships, node attributes and edge attributes. $ABAC_{NES}$ is comparable to $ReBAC_{BNES}$. This completes our discussion of Figure 6(a).

There are three types of ABAC in terms of possible changes in ABAC Models which we call ABAC dynamics. Figure 6(b) shows the dynamics as follows.

- **Static ABAC**: Nothing gets changed. In this type of ABAC, everything is static. Change of attribute values (i.e., assigning new values to attributes) or change of entities (i.e., adding or deleting entities) are not allowed.
- **Attribute Value Changes**: This ABAC allows changes of attribute values (assigning new values to attributes).

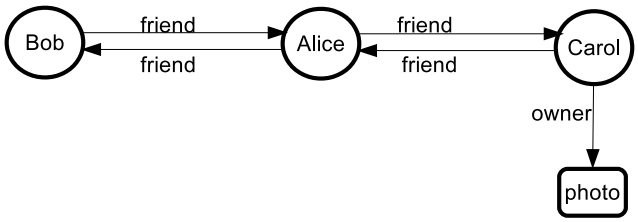

Figure 7: *A Simple Relationship Graph for Example 1*

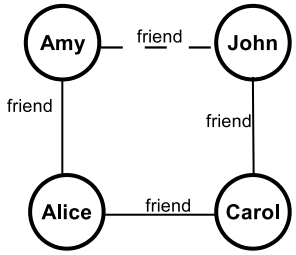

Figure 8: *A Simple Relationship Graph for Example 2*

- **Entity Changes**: This ABAC allows new entity creation and/or deletion. We understand that entity changes also includes attribute value changes as it needs assigning new values to attributes.

Each ABAC models shown in the Figure 6(a) can be combined with any dynamics shown in Figure 6(b).

6. EXPRESSING MULTILEVEL RELATIONSHIPS WITH ATTRIBUTES

Entity attributes can directly configure one level relationship such as parent, spouse, owner. Only entity attribute is allowed for attribute function composition. ReBAC is all about expressing authorization policy with multilevel or composite relationship (friend ∘ friend, friend ∘ parent etc.). In this subsection, we propose two methods of composite relationship expression using attributes.

1. **Attribute Composition or Chaining:** Attribute chaining is attribute function composition as defined in Section 3. Traditional ABAC uses direct attribute value of a user to specify policy. While attribute chaining approach allows to specify a policy through composition of attribute function. This approach requires runtime computation for relationship composition just like ReBAC.

2. **Composite Attribute:** In this approach, all possible or required paths of a relationship graph are captured as attributes. When an update occurs in the relationship graph, this approach needs to update attributes of directly and indirectly related entities. Here the term possible and required is used in the sense that the maximum possible depth of a graph depends upon its size while required depth means the limited depth required to specify authorization policy.

We discuss both concepts with some examples below.

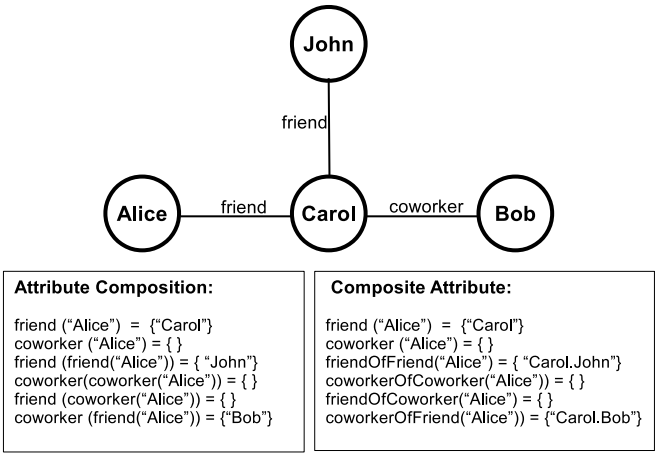

Figure 9: *Example of Attribute Composition and Composite Attribute in a Simple Relationship Graph (Example 3)*

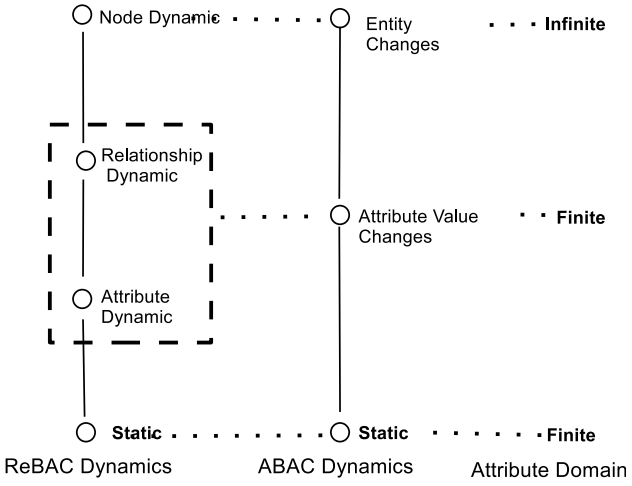

Figure 10: *Comparison Between ReBAC and ABAC with respect to Dynamics and Attribute Domain*

Example 1: Consider the relationship graph in Figure 7. Let's assume the policy for photo access allows only owner or owner's friend can access them.

Attribute Composition or Chaining : To configure this scenario with attribute composition approach, each user should have two entity attributes "friend" and "owner" and the authorization policy would check whether a particular user is in owner("photo") or friend(owner("photo")). According to this policy "Carol" and "Alice" can access "photo", but "Bob" cannot.

Composite Attribute: In this approach, to express the relationship graph and policy, ABAC should have user attributes, "friend" and "friendOfFriend", as well as object (photo) attributes, "owner", "friendOfOwner" and "friendOfFriendOfOwner". Here, "friendOfFriend", "friendOfOwner" and "friendOfFriendOfOwner" are composite attributes. The authorization policy would check whether a particular user is in owner("photo") or friendOfOwner("photo").

Here, owner("photo") = {"Carol"}, friendOfOwner("photo") = {"Alice"}, friendOfFriendOfOwner("photo") = {"Bob"}, fri-endOfFriend("Bob") = {"Carol"}, friendOfFriend("Carol") = { "Bob"}. If "friend" relationship between "Alice" and "Bob" is removed, it is necessary to update friend("Bob"), friend ("Alice") and friendOfFriend("Bob"). This action also requires indirect updates on friendOfFriend("Carol") and friend-OfFriendOfOwner("photo").

Example 2: Consider Figure 8 where "Alice" has friend "Carol" and "Amy". "Amy" and "Carol" both have a common friend "John". So "John" is Alice's friend ∘ friend through "Carol" and "Amy". Removing the relationship between "Amy" and "John" shouldn't remove "John" from "Alice"'s friendOfFriend list. This means, instead of simply storing friendOfFriend("Alice") = {"John"}, we need to store friend-OfFriend("Alice") = { "Amy.John", "Carol.John"}. Storing such path information as an attribute value would ensure availability of accurate attribute values. As demonstrated in this example, it is often not sufficient to store only the end user information as an attribute value in case composite attributes are used.

Example 3: Consider another example with the simple relationship graph shown in Figure 9.

Attribute Composition or Chaining: In this approach we need to have two entity attributes for users, "friend" and "coworker". To express a policy that verifies a composite relationship such as friend ∘ friend, coworker ∘ friend or friend ∘ coworker, we can use attribute composition such as friend(friend("Alice"))= {"John"}, coworker(friend("Alice")) = {Bob}, friend(coworker("Bob")) = {"John"}.

Composite Attribute: In this approach, we need to have "friend", "coworker", "friendOfFriend", "friendOfCoworker", "coworkerOfFriend" as attributes, so we can express relationship paths that might be found in policies without chaining attributes. This approach has maximum depth limit in expressing relationship based policy dependent on the attribute configuration. Every entity attributes defined in this approach should have a fixed relationship depth. For example "friend" and "coworker" express one level relationships while "friendOfFriend", "friendOfCoworker" and "coworkerOfFriend" express two level relationships.

7. COMPARISON: ABAC vs. ReBAC

In this section we compare ReBAC with ABAC, using the classifications of Sections 4 and 5. We conduct a conceptual comparison using two metrics: i) dynamics and ii) structural models. As the goal of this paper is to provide high level comparison, we do not provide any formally defined models or policy specifications. In order to use the formal framework of [50] to compare expressive power it is necessary to give detailed formal specifications of access control models. This limits comparison results to the very specific models that have been fully specified. We rather seek an intuitive but rigorous and insightful comparison between structurally comparable models.

In this work, we assume only entity attributes can configure relationships and non-entity attributes are finite domain attributes. We have shown that multilevel relationships can be configured with either attribute composition or with composite attributes. ReBAC node attributes can be configured using ABAC atomic or set-valued, and entity or non-entity attributes. ReBAC edge attributes can be configured using ABAC structured attributes of entities. From ReBAC point of view, if ABAC has only non-entity attributes, it means ReBAC graph structure has disconnected nodes with node attributes only. If ABAC has the capability to define en-

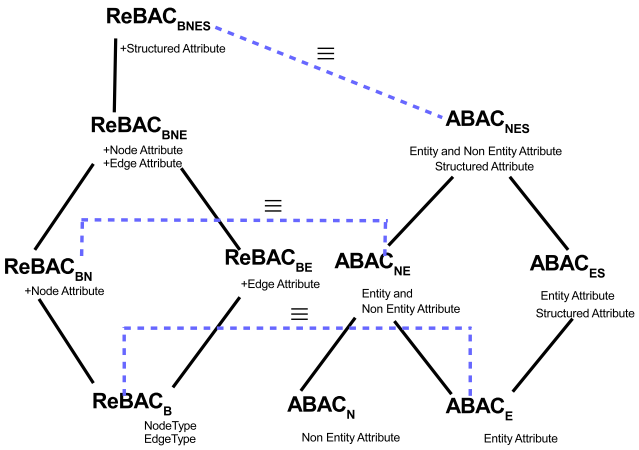

Figure 11: *Equivalence of ReBAC and ABAC Structural Classification*

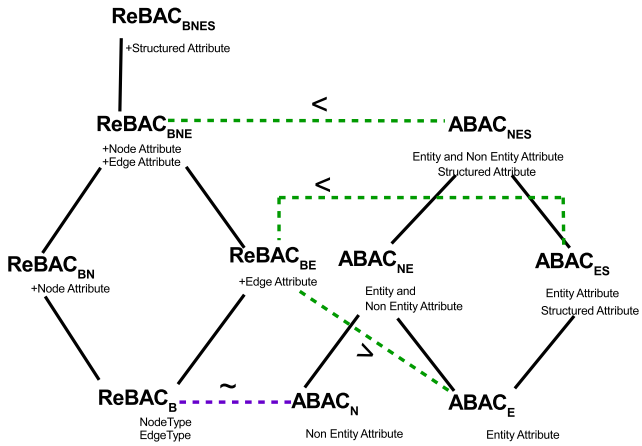

Figure 12: *Non-Equivalence of ReBAC and ABAC Structural Classification*

tity attributes, it can be configured to express relationships. Structured entity attributes can be configured as atomic or set-valued edge attributes or structured node attributes in relationship graph.

7.1 Comparison on Dynamics

Figure 10 shows a three-way alignment of ReBAC and ABAC dynamics with finite/infinite attribute domains. We understand this alignment to mean the following. The statement that ABAC_X is equivalent to ReBAC_Y is to be interpreted as given below.

- Static and finite attribute domain ABAC_X is equivalent to static ReBAC_Y.
- ABAC_X that allows change of attribute values with finite domain attribute is equivalent to relationship dynamic (which includes attribute dynamic where it is applicable) ReBAC_Y.
- ABAC_X that allows entity changes and infinite domain entity attribute is equivalent to node dynamic ReBAC_Y.

This alignment and interpretation allows us to avoid explicit consideration of all combinations of dynamics and models, which would be overwhelming. It does impose an obligation to consider all three levels of dynamics from Figure 10 in making equivalence claims.

We also have the following general result.

THEOREM 1. *Finite domain ABAC cannot configure ReBAC that changes entities in the relationship graph (i.e., node dynamic ReBAC).*

PROOF. (Sketch) Entity changes in ReBAC entail creating new entities in the system and deleting existing ones. In order to configure any kind of ReBAC we need entity attributes in ABAC. Changes of entity from ReBAC requires changing the range of entity attribute for ABAC to potentially unbounded size. A finite domain ABAC cannot have attributes that changes its range over time in this manner. □

7.2 Comparable Structural Models for ReBAC and ABAC

In this sub-section we compare the ReBAC and ABAC structural models from Figures 1(a) and 6(a) respectively. Figure 11 shows the equivalence of different ABAC and ReBAC models (with blue dotted lines) Figure 12 shows the non-equivalence of different ABAC and ReBAC models (purple dotted line shows one model is incomparable with another while green dotted line shows one model is more expressive than another).

THEOREM 2. *$ABAC_N$ is incomparable to $ReBAC_B$*

PROOF. (Sketch) ABAC_N has only non-entity attributes which cannot configure relations as discussed earlier. □

THEOREM 3. *$ABAC_E$ and $ReBAC_B$ are equivalent in expressive power.*

PROOF. (Sketch) To prove this we need to show

- ABAC_E can configure ReBAC_B
- ReBAC_B can configure ABAC_E

For the former, ABAC_E has entity attributes which can configure relationships via the techniques discussed in Section 6. For the latter, ABAC_E can be expressed as ReBAC_B where the entity attributes are relationship types, entities are nodes in the graph and which allows only one level relationship expression in authorization policy. □

COROLLARY 1. *$ABAC_N$ is incomparable to $ABAC_E$*

PROOF. (Sketch) Theorem 2 proves that ABAC_N is incomparable to ReBAC_B and Theorem 3 proves that ABAC_E and ReBAC_B are equivalent in expressive power. The corollary follows. □

THEOREM 4. *$ABAC_{NE}$ and $ReBAC_{BN}$ have equivalent expressive power*

PROOF. (Sketch) With entity attribute ABAC_NE can configure relationships of ReBAC_BN and with non-entity attribute ABAC_NE can configure non-entity node attribute of ReBAC_BN. So ABAC_NE can configure ReBAC_BN. Conversely ReBAC_BN can express entity attribute as relationships and non-entity attribute as node attribute in the relationship graph. So ReBAC_BN can configure ABAC_NE. □

THEOREM 5. *$ABAC_E$ is less expressive than $ReBAC_{BE}$*

PROOF. (Sketch) Entity attribute of $ABAC_E$ can be configured with relationship of $ReBAC_{BE}$. So $ReBAC_{BE}$ can configure $ABAC_E$. On the other hand we have seen in Section 5 that structured attributes are required to configure edge attributes in ABAC. For example consider Figure 4 where "tenantTrust" has "trustValue" as edge attribute. Without structured entity attribute, $ABAC_E$ cannot configure this example of $ReBAC_{BE}$. □

THEOREM 6. *$ABAC_{ES}$ is more expressive than $ReBAC_{BE}$*

PROOF. (Sketch) By definition $ABAC_{ES}$ has structured entity attribute while $ReBAC_{BE}$ does not have structured attributes. We have seen in section 5 with structured valued entity attribute $ABAC_{ES}$ can configure relationships, nodes and atomic or set-valued edge attribute of $ReBAC_{BE}$. So $ABAC_{ES}$ can configure $ReBAC_{BE}$. On the other hand $ReBAC_{BE}$ cannot configure more than one level structured entity attribute because it can have only atomic or set valued edge attribute. A 2-level structured entity attribute means at least one subattribute is also a structured attribute. So $ReBAC_{BE}$ cannot configure $ABAC_{ES}$. □

THEOREM 7. *$ABAC_{NES}$ is more expressive than $ReBAC_{BNE}$*

PROOF. (Sketch) Essentially similar proof as the previous theorem. □

THEOREM 8. *$ABAC_{NES}$ and $ReBAC_{BNES}$ have same expressive power*

PROOF. (Sketch) $ABAC_{NES}$ has structured entity and non-entity attributes while $ReBAC_{BNES}$ has labeled relationship graph (multiple type of relationships) with multiple types of nodes (entities) and structured node and edge attributes. Section 5 has shown that $ABAC_{NES}$ can configure relationships, nodes and structured attributes for nodes and edges. So $ABAC_{NES}$ can configure $ReBAC_{BNES}$. On the other hand $ReBAC_{BNES}$ can configure entities with nodes, structured entity and non-entity attributes with structured entity and non-entity node attributes respectively. So $ReBAC_{BNES}$ can configure $ABAC_{NES}$. This proves that $ABAC_{NES}$ and $ReBAC_{BNES}$ have same expressive power. □

7.3 Performance Comparison

So far we have considered the theoretical expressive power equivalence between ABAC and ReBAC. There are clearly some differences between them in terms of performance. ReBAC does runtime computation of authorization. Even if relationship graph is static and nothing changes, ReBAC still needs to repeat the same computation. To eliminate this massive redundant computation load researchers have considered caching of relationship paths [23]. In Section 6 we proposed two solutions for multilevel relationship expression in ABAC, viz., attribute composition and composite attributes. Attribute composition is similar to ReBAC in expressing policy, while composite attribute is more like caching of path information. Attribute composition has polynomial complexity for authorization policy and constant complexity for update, on the other hand composite attribute has constant complexity in policy authorization and polynomial time complexity on update to maintain relationship changes.

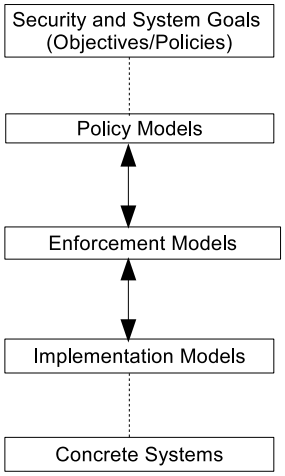

Figure 13: *PEI Framework [46]*

Performance also depends upon the characteristics of the system. A number of variances regarding system characteristics such as relationship dynamics, node dynamics and density of relationships between nodes (entities) affect performance. For meaningful performance comparison we need to formally define specific comparable models considering both approach, do their implementation and configure the system for different dynamics (attribute dynamics, node dynamics, relationship dynamics and density dynamics) variances.

7.4 Choices Of Models

Attribute composition or ReBAC approach puts the load on runtime computation, while caching or composite attribute may need significant update load. If relationship graph changes frequently, the caching or composite attribute approach needs to have excessive updates to keep the path information up-to-date.

The choice of models depends on node dynamics, relationship dynamics and the density of relationships between nodes (entities) in the system. If the relationship density of a system is high, adding or deleting a largely connected node will affect quite a large number of relationships in the system. For a static system or a system with non-entity attribute change, regardless of whether the graph is dense or sparse composite attribute is the best approach for relationship expression. If the system has huge node dynamics and relationship dynamics and relationship density is also high attribute composition would be the best solution. If the system is in the middle between these two extremes then we can think of an hybrid approach where both attribute composition and composite attribute are used in the same model. For example to achieve p level relationship composition we can use m level composite attribute and n level attribute composition where $p = n \times m$. To specify it more clearly we can say that a composite attribute with 4 level relationship expression capability such as ffff(u) or an attribute composition with 4 level relationship expression capability such as f(f(f(f(u)))) can be expressed with a composite attribute of 2 level relationship expression capability using 2 level attribute composition ff(ff(u)). This means ffff(u) = f(f(f(f(u)))) = ff(ff(u)).

Application context for security has the well established 3 layers (Policy P, Enforcement E and implementation I or PEI) [45, 46, 47], as shown in Figure 13. Policy level P is all about expressibility, modularity and convenience to express policy and independent of implementation detail. From expressibility point of view both the approaches are equal as we have already shown the equivalence of policy expression at the P layer. E layer is responsible for enforcement architecture wherein performance would come into consideration. Depending on the dynamics characteristics we conjecture that some hybrid combination of ABAC with attribute composition and composite attribute would be optimal for most situations.

8. CONCLUSION

In this paper we have provided an intuitive but rigorous comparative study of ABAC and ReBAC, and shown how various ReBAC features can be expressed with different types of ABAC. Our results indicate that the relationship between ABAC and ReBAC is subtle and variable depending on the precise flavor of these two access control approaches in any given model. At the same time we are able to make some general statements about this comparison. Additional work on comparing expressive power may yield additional insights. More significantly we believe metrics beyond theoretical equivalence need to be brought into consideration to better understand the relative advantages and disadvantages of these two approaches. Performance is one such metrics but others such as maintainability, robustness, and agility, also need to be studied.

Acknowledgements

This research is partially supported by NSF Grants CNS-1111925, CNS-1423481, CNS-1538418, and DoD ARL Grant W911NF-15-1-0518.

References

[1] Alloy language and tool. http://alloy.mit.edu/alloy/. Accessed 09/2016.

[2] OASIS, Extensible access control markup language (XACML), v2.0 (2005).

[3] Openstack. http://www.openstack.org/software/mitaka. Accessed 09/2016.

[4] Singlevalue multivalue. https://msdn.microsoft.com/en-us/library/aa746488(v=vs.85).aspx. Accessed 09/2016.

[5] Structured attribute. https://docops.ca.com/ca-identity-manager/12-6-5/en/configuring/user-console-design/configuring-profile-tabs-and-screens/field-styles/structured-attribute-display. Accessed 09/2016.

[6] Swift. http://docs.openstack.org/developer/swift/. Accessed 09/2016.

[7] T. Ahmed, F. Patwa, and R. Sandhu. Object-to-object relationship based access control: model and multi-cloud demonstration. In *IEEE Conference on Information Reuse and Integration (IRI)*. IEEE, 2016.

[8] M. Barr and C. Wells. Category theory for computing science. In *Prentice Hall*, page 6, 1998.

[9] P. Bennett, I. Ray, and R. France. Analysis of a relationship based access control model. In *Proceedings of the Eighth International C* Conference on Computer Science & Software Engineering*, pages 1–8. ACM, 2015.

[10] E. Bertino, P. Samarati, and S. Jajodia. An extended authorization model for relational databases. *IEEE Transactions on Knowledge and Data Engineering*, 9(1):85–101, 1997.

[11] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE symposium on security and privacy (SP'07)*, pages 321–334. IEEE, 2007.

[12] G. Bruns, P. W. Fong, I. Siahaan, and M. Huth. Relationship-based access control: its expression and enforcement through hybrid logic. In *ACM CODASPY*, pages 117–124, 2012.

[13] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham. A semantic web based framework for social network access control. In *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies*, SACMAT '09, pages 177–186, New York, NY, USA, 2009. ACM.

[14] B. Carminati, E. Ferrari, and A. Perego. Rule-based access control for social networks. In *OTM Confederated International Conferences: On the Move to Meaningful Internet Systems*, pages 1734–1744. Springer, 2006.

[15] B. Carminati, E. Ferrari, and A. Perego. Enforcing access control in web-based social networks. *ACM Transactions on Information and System Security (TISSEC)*, 13(1):6, 2009.

[16] D. Chadwick. *Understanding X.500: The Directory*. Chapman & Hall, Ltd., London, UK, 1994.

[17] M. Chase. Multi-authority attribute based encryption. In *Theory of Cryptography Conference*, pages 515–534. Springer, 2007.

[18] P. P.-S. Chen. The entity-relationship model toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, 1(1):9–36, 1976.

[19] Y. Cheng, K. Bijon, and R. Sandhu. Extended ReBAC administrative models with cascading revocation and provenance support. In *Proceedings of the 21st ACM on Symposium on Access Control Models and Technologies*, pages 161–170. ACM, 2016.

[20] Y. Cheng, J. Park, and R. Sandhu. Relationship-based access control for online social networks: Beyond user-to-user relationships. In *International Conference on Privacy, Security, Risk and Trust (PASSAT)*, pages 646–655. IEEE, 2012.

[21] Y. Cheng, J. Park, and R. Sandhu. A user-to-user relationship-based access control model for online social networks. In *Data and applications security and privacy XXVI*, pages 8–24. Springer, 2012.

[22] Y. Cheng, J. Park, and R. Sandhu. Attribute-aware relationship-based access control for online social networks. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 292–306. Springer, 2014.

[23] J. Crampton and J. Sellwood. Caching and auditing in the RPPM model. In *International Workshop on Security and Trust Management*, pages 49–64. Springer, 2014.

[24] J. Crampton and J. Sellwood. Path conditions and principal matching: a new approach to access control. In *Proceedings of the 19th ACM symposium on Access control models and technologies*, pages 187–198. ACM, 2014.

[25] J. Crampton and J. Sellwood. ARPPM: Administra-

tion in the RPPM model. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pages 219–230. ACM, 2016.

[26] J. Crampton and J. Sellwood. Inter-ReBAC: interoperation of relationship-based access control model instances. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 96–105. Springer, 2016.

[27] R. Fagin. On an authorization mechanism. *ACM Transactions on Database Systems (TODS)*, 3(3):310–319, 1978.

[28] P. W. Fong. Relationship-based access control: protection model and policy language. In *Proceedings of the first ACM conference on Data and application security and privacy*, pages 191–202. ACM, 2011.

[29] P. W. Fong, M. Anwar, and Z. Zhao. A privacy preservation model for facebook-style social network systems. In *Computer Security–ESORICS 2009*, pages 303–320. Springer, 2009.

[30] P. W. Fong and I. Siahaan. Relationship-based access control policies and their policy languages. In *Proceedings of the 16th ACM symposium on Access control models and technologies*, pages 51–60. ACM, 2011.

[31] J. Gallier. Discrete mathematics. In *PWS Publishing*, page 118. Springer, 2011.

[32] P. P. Griffiths and B. W. Wade. An authorization mechanism for a relational database system. *ACM Transactions on Database Systems (TODS)*, 1(3):242–255, 1976.

[33] V. C. Hu, D. Ferrariolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and S. Karen. Guide to attribute based access control (ABAC) definitions and considerations. In *NIST Special Publication 800-162*, SIN '13, 2014.

[34] X. Jin. *Attribute-Based Access Control Models and Implementation in Cloud Infrastructure as a Service*. PhD thesis, UTSA, 2014.

[35] X. Jin, R. Krishnan, and R. Sandhu. A unified attribute-based access control model covering DAC, MAC and RBAC. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 41–55. Springer, 2012.

[36] J. Kolter, R. Schillinger, and G. Pernul. A privacy-enhanced attribute-based access control system. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 129–143. Springer, 2007.

[37] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. volume 24, pages 131–143. IEEE, 2013.

[38] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 195–203. ACM, 2007.

[39] J. Pang and Y. Zhang. A new access control scheme for facebook-style social networks. *Computers & Security*, 54:44–59, 2015.

[40] J. Park and R. Sandhu. The UCONabc usage control model. *ACM Trans. Inf. Syst. Secur.*, 2004.

[41] B. Qin, H. Deng, Q. Wu, J. Domingo-Ferrer, D. Naccache, and Y. Zhou. Flexible attribute-based encryption applicable to secure e-healthcare records. volume 14, pages 499–511. Springer, 2015.

[42] S. Z. R. Rizvi and P. W. Fong. Interoperability of relationship-and role-based access control. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pages 231–242. ACM, 2016.

[43] S. Z. R. Rizvi, P. W. Fong, J. Crampton, and J. Sellwood. Relationship-based access control for an open-source medical records system. In *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies*, pages 113–124. ACM, 2015.

[44] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–473. Springer, 2005.

[45] R. Sandhu. Engineering authority and trust in cyberspace: The om-am and rbac way. In *Proceedings of the fifth ACM workshop on Role-based access control*, pages 111–119. ACM, 2000.

[46] R. Sandhu. The PEI framework for application-centric security. In *Security and Communication Networks (IWSCN), 2009 Proceedings of the 1st International Workshop on*, pages 1–6. IEEE, 2009.

[47] R. Sandhu, K. Ranganathan, and X. Zhang. Secure information sharing enabled by trusted computing and PEI models. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 2–12. ACM, 2006.

[48] H. Shen. A semantic-aware attribute-based access control model for web services. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 693–703. Springer, 2009.

[49] S. D. Stoller. An administrative model for relationship-based access control. In *Data and Applications Security and Privacy XXIX*, pages 53–68. Springer, 2015.

[50] M. V. Tripunitara and N. Li. A theory for comparing the expressive power of access control models1. volume 15, pages 231–272. IOS Press, 2007.

[51] E. Yuan and J. Tong. Attributed based access control (ABAC) for web services. In *Proceedings of the IEEE International Conference on Web Services*, ICWS '05, pages 561–569, Washington, DC, USA, 2005. IEEE Computer Society.