Current Research and Open Problems in Attribute-Based Access Control

DANIEL SERVOS and SYLVIA L. OSBORN, University of Western Ontario

Attribute-based access control (ABAC) is a promising alternative to traditional models of access control (i.e., discretionary access control (DAC), mandatory access control (MAC), and role-based access control (RBAC)) that is drawing attention in both recent academic literature and industry application. However, formalization of a foundational model of ABAC and large scale adoption is still in its infancy. The relatively recent emergence of ABAC still leaves a number of problems unexplored. Issues like delegation, administration, auditability, scalability, hierarchical representations, and the like, have been largely ignored or left to future work.

This article provides a basic introduction to ABAC and a comprehensive review of recent research efforts toward developing formal models of ABAC. A taxonomy of ABAC research is presented and used to categorize and evaluate surveyed articles. Open problems are identified based on the shortcomings of the reviewed works and potential solutions discussed.

CCS Concepts: • **Security and privacy → Access control;**

Additional Key Words and Phrases: Attribute-based access control (ABAC), access control, ABAC models, survey

ACM Reference Format:
Daniel Servos and Sylvia L. Osborn. 2017. Current research and open problems in attribute-based access control. ACM Comput. Surv. 49, 4, Article 65 (January 2017), 45 pages.
DOI: http://dx.doi.org/10.1145/3007204

1. INTRODUCTION

Attribute-Based Access Control (ABAC) is an emerging form of access control that is starting to garner interest in both recent academic literature and industry application. While there is currently no single agreed upon model or standardization of ABAC, there are commonly accepted high level definitions and descriptions of its function. One such high level description is given in National Institute of Standards and Technology (NIST)'s recent publication, a "Guide to Attribute Based Access Control (ABAC) Definition and Considerations" [Hu et al. 2013]:

> ***Attribute-Based Access Control:*** *An access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environmental conditions, and a set of policies that are specified in terms of those attributes and conditions.*

ABAC, unlike more traditional models of access control, allows for the creation of access policies based on the existing attributes of the users and objects in the system, rather than the manual assignment of roles, ownership, or security labels by a system

Authors' addresses: D. Servos and S. L. Osborn, Department of Computer Science, Middlesex College, Western University, London, Ontario, Canada, N6A 5B7; emails: dservos5@uwo.ca, sylvia@csd.uwo.ca.

DOI: http://dx.doi.org/10.1145/3007204

administrator. There are several situations, including cloud computing, where this would be beneficial, removing the need for manual intervention when authorizing users for certain roles or security levels, simplifying administration in complex systems with a large number of users, as well as creating the possibility of automating access control decisions for remote users from foreign systems.

While many works have explored the application of ABAC to existing problems and have attempted to further formalize ABAC, few have sought to provide an in-depth summary of current efforts or detail the open problems present in the area of ABAC research. This article seeks to provide such a summary and to identify open problems currently limiting real-world implementation and use of ABAC. We introduce a taxonomy of current areas of ABAC research, provide a survey and review of the most notable works to date, and detail some of the most pressing open problems.

The remainder of this article is divided into the following sections: Section 2 gives a brief background on ABAC and introduces the "Core" ABAC Model, Section 3 describes the methodology used for choosing the papers and works surveyed, Section 4 provides a taxonomy of current areas of ABAC research, and Section 5 reviews the most notable ABAC models and frameworks. Finally, Section 6 identifies and discusses open problems not yet addressed by present ABAC efforts, while Section 7 provides concluding remarks.

2. ABAC BACKGROUND

Rather than basing access control decisions on a user's identity, like the traditional methods, ABAC bases access control on the attributes of access control entities. These attributes are often classified into one of the following categories:

- ***User Attributes.*** Attributes of the subjects of the system. May include attributes like age, name, office number, job title, role, security clearance, home address, date hired, trust level (e.g., how trusted the user is by the system), and so on.
- ***Object Attributes.*** Attributes of the resources of the system. May include attributes about the meta-data related to the object such as author, date created, last modified, size, file type, security level, and so forth, or the contents of the object such as patient name (e.g., for health records), student number (e.g., for student records), title of Chapter 1, and the like.
- ***Environmental Attributes.*** Attributes derived from the current state of the system's environment. For example, current time, day of the week, number of users logged in, free space, CPU usage, and so on.
- ***Connection Attributes.*** Attributes that only apply to the current session of a user. For example, IP address, physical location (e.g., for mobile systems), session start date/time, current session length, host name, number of access requests made, and so forth.
- ***Administrative Attributes.*** Configuration attributes that apply to the whole system and are either manually set by an administrator or by some automated process. These could include a threat level (e.g., different policies could be used depending on whether or not the system was likely to be attacked), minimum trust level (e.g., the minimum amount of trust required for a user to access the system), maximum session length (e.g., the maximum allowable length of a session), and the like.

Ideally, these attributes are all properties of the elements in the system and do not need to be manually entered by administration (e.g., many of the attributes about an object come from its meta-data). Access policies can be created using policy languages, limiting access to certain resources or objects, based on the result of a Boolean statement

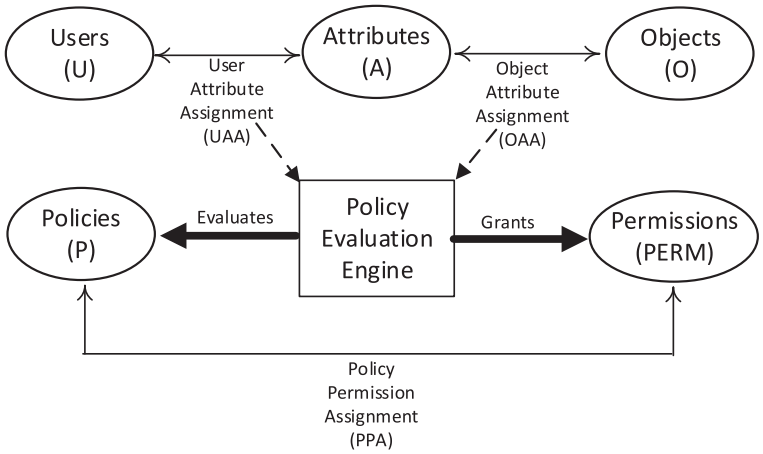


Fig. 1. Core ABAC model. Thin solid arrows denote many-to-many relations, thick solid lines denote relation with policy engine, and dotted lines denote information used by the policy engine to evaluate a given policy. Ovals represent ABAC model elements.

comparing attributes, for example *"user.age >= 18 OR object.owner == user.id"* or *"TIME > 8:00AM AND TIME < 5:00PM."* This allows for flexible enforcement of real-world policies, while only requiring knowledge of some subset of attributes about a given user (as opposed to knowing their identity and to what roles or permissions they have been manually assigned).

2.1. Core ABAC Model

This section gives a description of a simplified ABAC model based on common elements found in most ABAC models. While each ABAC model tends to formalize the elements of ABAC in a slightly different way, the following are the most common elements of an ABAC system and are present in most models:

Users (U). The set of all users that may access the system. Note that this set may not necessarily be finite as not all users are known at creation time (something that is common in service oriented architectures and systems involving information sharing across organizational boundaries).

Objects (O). The set of all objects protected by the system.

Attributes (A). The set of all attributes (given by a unique name) in the system. In some models, attributes also have a type associated with them or are subdivided into categories based on the access control entity to which they can be applied.

Permissions (PERM). The set of all possible permissions that may be granted to users. In some models, permissions consist of object, operation pairs similar to permissions in role-based access control (RBAC), but this is not necessarily required. In other models, permissions are policy and operation pairs that grant access to execute the operation on any object that fulfils the policy.

Policies (P). The set of all policies that govern access in the system. Normally, these policies are written in a policy language and in some way related to permissions they grant.

Users and objects are assigned attributes and related through the following relations (shown in Figure 1):

Users Attribute Assignment (UAA). The assignment of attributes to users. This may take the form of $\{a \in A, u \in U, values\} \in UAA$; that is to say that each

element of UAA is a triple containing an attribute name from the set of attributes (A), a user from the set of users (U), and a set of values assigned to the given user and attribute pair. For example, if a user, u_1, was assigned an "age" attribute with the value of 28, the entry in UAA would be {"*age*", u_1, {28}}. Alternatively, if a user, u_2, was assigned a "supervises" attribute that contains the set of other users they supervise (in this case, u_1 and u_3), the entry in UAA would be {"*supervises*",u_2, {u_1, u_3}}.

Object Attribute Assignment (OAA). The assignment of attributes to objects. This may take the form of $\{a \in A, o \in O, values\} \in OAA$, and works in the same way as UAA but with objects.

Policy Permission Relation (PPR). The relationship between policies and the permissions they grant. This may take the form of $\{p \in P, perm \subseteq PERM\} \in PPR$. This assignment is often formulated differently or not at all in many models, depending on how their policy language works (e.g., the language itself may specify the permission set granted).

Policies in the P set are commonly Boolean statements involving attributes and constants such as *"user.age >= 18"* (grants access if the user is 18 or more years of age) or *"user.id == object.author"* (grants access if the user is the author of the file). When an access request is made by a user, it is evaluated against the set of policies (P) given the assigned attributes of the user making the request and the object being requested. In many models, access requests are not conducted directly by the user but indirectly through a session that may contain a subset of the user's attributes. A comprehensive review of existing ABAC models is given in Section 5.

2.2. Policy Language Standards

A critical component of ABAC, although not strictly part of the ABAC model, is the access control policy language used to define policy rules for a system. These languages, while not models in themselves (as is sometimes erroneously implied), are either generic access control language standards (such as XACML) or languages created specifically for use with a single model. eXtensible Access Control Markup Language (XACML) [Godik et al. 2002], a standard created by the Organization for the Advancement of Structured Information Standards (OASIS), is one of the most frequently referenced works in ABAC literature. XACML is an XML-based access control policy language that is notable for its support of attribute-based policies and used in multiple access control products. Similarly, Security Assertion Markup Language (SAML) [Hughes and Maler 2005], also developed by OASIS, provides a standardized markup language and protocol for exchanging attribute-based authorization and authentication information between service providers, identity/attribute providers, and users.

2.3. Attribute-Based Encryption

Another related but distinct research area from ABAC is attribute-based encryption (ABE), where objects are encrypted based on attribute-based access policies. ABE mainly consists of key-policy ABE (KP-ABE) [Goyal et al. 2006] or ciphertext-policy (CP-ABE) [Bethencourt et al. 2007; Servos et al. 2013] based encryption ciphers. In KP-ABE, an object is encrypted with a set of attributes related to the object that must pass a policy embedded in a user's key for decryption to proceed. CP-ABE is the reverse of KP-ABE, using an attribute-based policy to encrypt an object and having a user's key consist of a set of attributes relating to that user. While ABE, much like XACML and SAML, lacks any kind of formal ABAC model and has rather simplified access policies, it does provide an interesting means of enforcing ABAC policies outside of the

security domain they originate in. There are several examples of ABE being used for such in recent literature [Hur and Noh 2011; Wang et al. 2010; Servos 2012; Yu et al. 2010; Bobba et al. 2010], particularly for securing web- and cloud-based services.

3. METHODOLOGY

A structured approach was used to locate peer-reviewed literature related to ABAC for the purposes of this literature survey. Searches for refereed journal papers, conference papers, and dissertations were conducted using the Google Scholar (http://scholar.google.ca) and DBLP (http://dblp.org/search/) search engines with queries relating to ABAC (e.g., searching for paper titles containing "attribute-based access control," "ABAC," "attribute-based"). Articles were then manually reviewed for inclusion/exclusion based on the following criteria:

Inclusion Criteria:
Papers and articles discussing models, implementations, frameworks, and architectures involving ABAC were included. Works dealing with attribute-based policies and policy languages were also included as well as works describing attribute sharing, storage, and privacy but are not discussed in this article beyond their inclusion in the taxonomy in Section 4 and statistics given in this document.

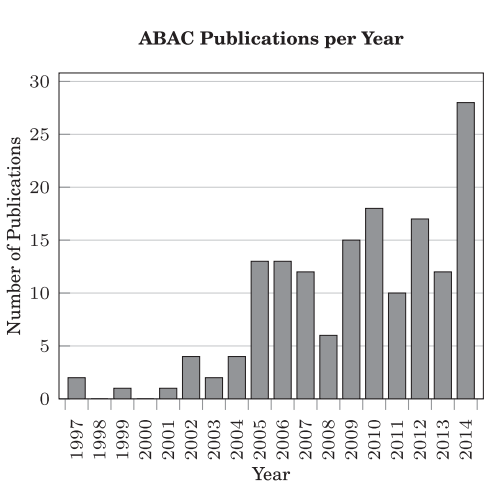
Exclusion Criteria:

—Any non-refereed work including patents, standards (XACML and SAML are mentioned due to their frequent mention in refereed literature, but not included in the statistics in this document), technical reports, or special publications (The NIST Guide to Attribute-Based Access Control (ABAC) Definition and Considerations is discussed in the introduction of this document but not included in the statistics).
—Any work that is not in English or is incomprehensible due to language issues (e.g., poorly translated articles).
—No documents were intentionally excluded based on date of publication.
—Any literature related to, or primarily using ABE was excluded as this literature search is intended to focus on models, frameworks, architectures, and use of ABAC, as opposed to attribute-base cryptography. While ABE may be a useful tool for enforcing attribute-based policies in environments where traditional policy enforcement is not possible (e.g., in off-line or untrusted environments), it in itself does not provide an underlying model for access control and only comprises one component of a complete security architecture.
—Any article that was superseded by another work by the same authors is excluded for the newer work. For example, if an author published the beginnings of an ABAC model in a conference and then further developed and finalized this model in a later journal paper, both works are considered to be the same model (both are included in the statistics in Figures 2 and 3).

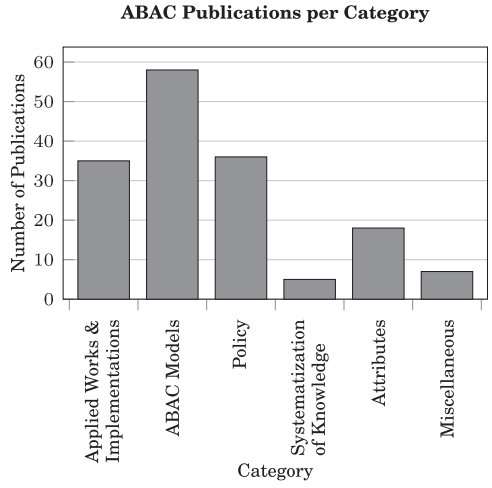
The result of this manual search found 199 papers that fall into at least one of the categories described in Section 4. A summary of the year in which each paper was published is given in Figure 2(a) and the category to which it belongs in Figure 2(b). From this set of papers, the most notable and relevant from the category of "ABAC Models" and its child categories are reviewed in Section 5 in this article.

4. TAXONOMY OF CURRENT AREAS OF RESEARCH

The current body of ABAC related research can be classified into a number of hierarchical categories as described in Table I. This taxonomy of current ABAC research was created after manually analyzing the peer-reviewed literature found via the methodology described in Section 3 and grouping works describing similar aspects of

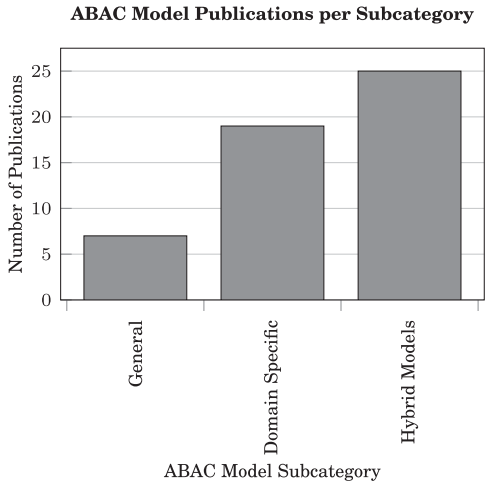


(a) ABAC publications per year between January 1st, 1997 and October 20th, 2014.

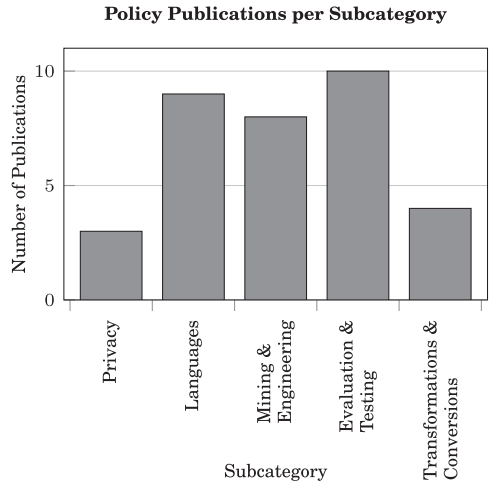


(b) ABAC publications per category as defined in Section 4.

Fig. 2. Distribution of ABAC related works matching criteria outlined in Section 3.



(a) ABAC model publications per subcategory as defined in Section 4.



(b) Policy publications per subcategory as defined in Section 4.

Fig. 3. Distribution of ABAC-related works matching criteria outlined in Section 3 (continued from Figure 2).

ABAC together. Related groups (e.g., Policy Languages and Policy Mining) were then further grouped under a more general category (e.g., Policy) that adequately describes all members of the child groups. A diagram of the taxonomy is presented in Figure 5 in the Appendix of this article.

5. MODELS AND FRAMEWORKS

5.1. Pure ABAC Models

Recent efforts have aimed to take the first steps toward creating foundational models of "pure" ABAC (i.e., ABAC models that are not simply extensions to existing models,

Table I. Description of Taxonomy Categories in Figure 5

Category	Description
Applied Works & Implementations	Literature describing implementations of ABAC systems, frameworks for using XACML, SAML, and the like, or any kind of application of existing ABAC research.
• **XACML-Based**	Implementations or applied work using XACML.
• **SAML-Based**	Implementations or applied work using SAML.
• **Other**	Any literature describing implementations or applied work that does not fit into the above subcategories.
ABAC Models	Literature describing access control models that incorporate attributes into access control decisions.
• **HybridModels**	Models that extend or combine existing (non-ABAC) models of access control (e.g., RBAC) to incorporate attributes.
–PRBAC	Parameterized Role-Based Access Control (PRBAC) models are RBAC models based around extending RBAC by parameterizing permissions and/or roles as described in Section 5.2.1.
–Attribute-Based Role Assignment	Models that extend RBAC to add attributes as described in Kuhn et al.'s Dynamic Roles strategy (i.e., assigning roles via user attributes). Described in Section 5.2.2.
–Attribute-Centric	Models that extend RBAC to add attributes as described in Kuhn et al.'s Attribute-Centric strategy that would not be classified as "pure" models of ABAC. Described in Section 5.2.3.
–Role-Centric	Models that extend RBAC to add permission filtering based on attributes as described in Kuhn et al.'s Role-Centric strategy. Described in Section 5.2.4.
–Unified Models	Access control models that combine ABAC with with alternative access control models (i.e., non-traditional models) as described in Section 5.2.5.
• **PureABACModels**	ABAC models that are not extensions to existing models of access control but new attribute-based models.
–General	ABAC models that are system independent in that they are general enough to be applied to any access control use.
–Domain Specific	ABAC models that are designed for a particular domain or use (e.g., for protecting web services).
* **Cloud Computing**	Models targeting the domain of cloud computing.
* **Real-time Systems**	Models targeting the domain of real-time systems.
* **Collaborative Environments**	Models targeting the domain of collaborative work and educational environments.
* **Mobile Environments**	Models targeting the domain of mobile environments, including both systems that track mobile physical objects and mobile computing systems (e.g., cell phones).
* **Grid Computing**	Models targeting the domain of grid computing.
* **Web Services**	Models targeting the domain of web services, including service-oriented architectures.
* **Other**	Any domain specific model that does not fit in one of the above child categories.
Policy	Literature describing the mining for or evaluation, testing, and development of attribute-based policies and languages. Also includes works attempting to preserve the privacy of policies or otherwise hide details of policies from an adversary.
• **Confidentiality**	Works aimed at preserving the privacy of attribute-based policies or otherwise hide details of policies from an adversary.
• **Languages**	Literature describing or extending attribute-based policy languages.
• **Mining & Engineering**	Research aimed at the automatic mining of attribute-based policies or otherwise engineering attribute-based policies.
• **Evaluation & Testing**	Literature describing the testing and evaluation of attribute-based policies. Includes both the implementation of tools to automate the testing of policies and efforts to prove the security/safety of policies (formally or otherwise).
Systematization of Knowledge	Literature reviews and systematization of knowledge in the area of ABAC.
Attributes	Works relating to sharing, storing, validating, securing, or ensuring the privacy of attributes used in ABAC.
• **Confidentiality**	Efforts to ensure the privacy of attributes. That is, protecting unwanted entities from determining the value of potentially sensitive attributes.
• **Storage & Sharing (Certificates)**	Efforts to enable the sharing or storage of attributes. Includes frameworks, protocols, and data structures (e.g., attribute certificates) for securely sharing attributes between access control entities.

e.g., RBAC, but new attribute-based models that can be seen as a generalization of traditional models). A summary of the most relevant attempts at creating such a model are given in Tables III and IV in the Appendix, with a more in-depth review of each being given later in this section. These efforts can be subdivided into two categories (as described in Section 4 and Figure 5), "general" and "domain specific." "Domain specific" models aim to provide ABAC for specific use cases such as cloud computing, web services, and the like, while "general" models aim to provide an ABAC solution that may be applied to any situation where access control is desired.

5.1.1. General Models.

A Logic-Based Framework for Attribute-Based Access Control. Wang et al. put forth one of the first "pure" and "general" ABAC models (published in 2004) in the form of a logic-based framework based on logic programming where policies are specified as "stratified constraint flounder-free logic programs that admit primitive recursion" [Wang et al. 2004] and attributes and operations are modeled as sets in computable set theory [Dovier et al. 2000]. Methods of optimizing the runtime performance of evaluating an ABAC-based policy are also demonstrated, which involve transforming a given ABAC policy into a semantically equivalent but runtime and overhead reduced policy when possible. While Wang et al.'s framework introduces hierarchical attributes (something lacking from other models), it is largely focused on the representation, consistency, and performance of attribute-based policies and their evaluation. Several critical components are absent, including lacking object attributes (the only attributes considered are user attributes) and omitting formalization of ABAC aspects outside of policies and their evaluation (e.g., only access control on services/operations is considered).

Attribute-Based Access Matrix Model. The paper by Zhange et al. [2005] proposes a unique model of ABAC based around an attribute enhanced access matrix, called the "attribute-based access matrix" (ABAM) model. ABAM defines an access matrix in which each row is represented by a pair consisting of a subject and its set of attributes $(S_i, ATTS(S_i))$ and each column by a pair consisting of an object and its set of attributes $(O_i, ATTS(O_i))$. Each cell $([S_i, O_i])$ then corresponds to the set of access rights the subject (S_i) may exercise over the object (O_i) assuming certain policies are fulfilled. Operations (or "commands" as they are called in ABAM) may be executed by a given subject over a given object only if the matching access rights required by the operation are found in the access matrix and the subject and object's attributes fulfill the set of policies on the operation.

In addition to the formalization of ABAM, Zhang et al. also provide a safety analysis to prove the decidability of ABAM for a case where the set of attributes is finite, and the attribute relationships allow no cycles. While ABAM's unique use of an access matrix allows for a more auditable ABAC system than other models (basic checks on which users may access a certain object may be accomplished with a simple matrix lookup rather than evaluating policies on a large set of attributes and subjects), it omits details on how policies are administered, composed, or evaluated. A policy language is shown in examples but never formalized fully. Similarly, it is stated that ABAM is comprehensive enough to encompass the traditional access control models; however, this is not demonstrated and it is left unclear how ABAM might encompass mandatory access control (MAC) or hierarchical RBAC. Lastly, ABAM lacks connection, environment, and hierarchical attributes as well as constraints to enforce separation of duty or enable delegation.

Secure Collaborations with Attribute-Based Access Control. A more recent work (2013) by Rubio-Medrano et al. [2013] introduces the notion of security tokens into

an abstract model of ABAC that defines the relevant core components and attributes required for a minimal reference model. Unlike other rule-based ABAC models that make access control decisions on the basis of evaluating policies given the current state of various attributes, Rubio-Medrano et al.'s model maps attributes of access control entities (subjects, objects, etc.) to security tokens by traversing an administrator defined "token-provisioning graph" (TP-Graph). The TP-Graph is a directed, possibly cyclic, graph whose vertices represent sets of related attributes or security tokens (referred to as attribute or security token families) and its edges represent "token-provisioning functions" (TP-Functions) that map attribute or security token families to a different security token family based on defined criteria the attribute or token values must meet. By allowing system administrators to define TP-Functions and relating security tokens to the permissions (object, operation pairs) they grant, it enables access control decisions that are claimed to be more auditable and open to security analysis using techniques based on graph theory.

While Rubio-Medrano et al.'s model gives a novel take on ABAC, the added auditability and graph-based security analysis come at the cost of increased administrative complexity and overhead. In theory, the TP-Graphs should allow for the development of security analysis techniques based on graph theory, but this seems to be largely left to future works. Additionally, the ABAC model itself is largely informal, leaving most concepts well-described but not defined formally. It is left unclear how TP-Functions and the TP-Graph may be created by an administrator or in what form they may take (a policy language is hinted at when directions for future work are discussed). Similarly, no precise description or algorithm is given for how the TP-Graph is traversed or how cycles may be handled.

ABAC$_\alpha$. Another recent (2012) work by Jin et al. [2012a] aims "to develop a formal ABAC model that is just sufficiently expressive to capture DAC, MAC, and RBAC". This model, *ABAC$_\alpha$*, provides formalizations of the basic ABAC elements (users, objects, policies, etc.), their relations, and constraints that allow emulation of the traditional models. A partial policy and constraint language, called "Common Policy Language (CPL)", based on set theory notation and Boolean logic, is defined and example configurations are given for discretionary access control (DAC), MAC, and RBAC-style access control in *ABAC$_\alpha$*. Additionally, a limited functional specification, including a bare minimum of administrative functions is specified (although details on what authorization conditions may be required for administrative functions are not given).

CPL is used for both policy specification and configuring constraints on *ABAC$_\alpha$* to limit possible attribute assignments and set a valid range and type of attribute values. Example 1 shows an authorization policy in CPL for enforcing RBAC style access control. In this case, *S* is the set of all subjects, *O* is the set of all objects, *srole* is a subject attribute that contains the subjects roles, *rrole* is an object attribute that contains the set of roles that grant permission to read the object, and *wrole* is an object attribute that contains the set of roles that grant permission to write to the object. The authorization policy states that a subject can only read the object if they have a role in the objects *rrole* attribute value set and can only write to the object if they have a role in the objects *wrole* attribute value set.

Example 1. Simple (non-hierarchical) RBAC authorization policy:

$$\texttt{Authorization}_{\texttt{read}}\texttt{(s : S, o : O)} \equiv \exists \texttt{r} \in \texttt{srole(s)} \in \texttt{rrole(o)}$$
$$\texttt{Authorization}_{\texttt{write}}\texttt{(s : S, o : O)} \equiv \exists \texttt{r} \in \texttt{srole(s)} \in \texttt{wrole(o)}$$

While this work provides a sufficient basis on which new foundational models of ABAC may feasibly be built, it (intentionally) lacks several necessary components for the real world. Features such as attribute and object hierarchies, environment and

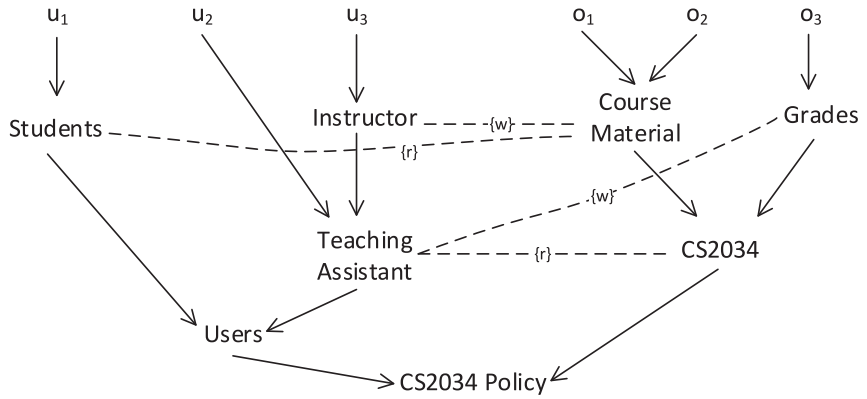


Fig. 4. Example Policy Machine policy class. Solid arrows represent attribute assignments, while dashed lines represent capabilities of the shown user attributes.

connection attributes, delegation, and separation of duties are omitted and left to future models built upon $ABAC_\alpha$. Finally, the given policy language, while adequate for modeling traditional access control systems, is insufficient for real-world application. No specifics are given on how CPL might handle multiple policy composition or conflicting policies and the heavy use of set theory notation in the language (as opposed to traditional Boolean statements) makes CPL's practicality over an existing policy language such as XACML questionable (creating XACML profiles for $ABAC_\alpha$ is left to future works).

The Policy Machine. Ferraiolo et al. [2011, 2015] have developed a novel approach to access control that is highly attribute-based in the form of the Policy Machine (PM). The PM is an architecture and access control framework to support the specification and enforcement of attribute-based access control policies that aims to redefine and generalize access control to provide a unified mechanism under which a wide range of policies may be enforced. Unlike other approaches that define attributes as name value pairs, the PM represents user attributes as many-to-many relations between users and capabilities (operation object pairs that grant the ability to perform the given operation on the given object). Similarly, object attributes are defined as many-to-many relations between sets of objects and sets of access entries (user operation pairs that state that the given user may perform the given operation). Attributes are hierarchical, allowing attributes to be assigned to other attributes so long as the chain of assignments remains acyclic. If two user attributes ua_1 and ua_2 exist such that ua_1 is assigned to ua_2, the set of users assigned to ua_1 are contained in ua_2 and the capabilities granted from ua_1 are those obtained through the chain of attribute assignments (e.g., all users assigned to ua_1 in this case would gain the capabilities granted from both ua_1 and ua_2). Assignments between object attributes work in a similar manner. If two object attributes oa_1 and oa_2 exist such that oa_1 is assigned to oa_2, the set of objects assigned to oa_1 are contained in oa_2 and the objects of oa_1 have the access entries assigned to oa_2 (in addition to those assigned to oa_1).

Policies are specified using policy classes, chains of attribute assignments terminating with a policy class as shown in the example policy given in Figure 4. In this example, an RBAC style policy class is shown that governs access to materials and grades for a university course. The user attribute *Instructor* grants the capability to write to objects assigned the *Course Material* attribute (in this case, o_1 and o_2); however, as the *Instructor* attribute is assigned the *Teaching Assistant* attribute, it also grants the capabilities of the *Teaching Assistant* attribute (and all other user attributes

on the path to the policy class in Figure 4). The *Teaching Assistant* attribute grants the capability to read all objects assigned with the *CS2034* attribute. This includes any objects assigned attributes that are in turn assigned the *CS2034* attribute (i.e., the *Course Material* and *Grades* object attributes) in the assignment chain. In this example, the resulting permissions allow teaching assistants (i.e., u_2) to read all of the CS2034 objects (o_1, o_2, and o_3) but only write to the grade objects (o_3). Instructors (i.e., u_3) have all permissions of teaching assistants in addition to being able to write to Course Material objects (o_1 and o_2). Finally, Students (i.e., u_1) are limited to only reading Course Material objects (o_1 and o_2).

Ferraiolo et al. show that the PM is sufficiently flexible to enforce DAC, MAC, RBAC and Chinese Wall [Brewer and Nash 1989] style security policies and provide further means to constrain policies with prohibitions, restrictions, and obligations. An administration model is also presented, as well as details on a number of architectural components necessary for implementation. The PM specification described in Ferraiolo et al. [2015] has served as the basis for the ANSI/INCITS Next Generation Access Control standardization effort [INCITS 2013, 2015].

Hierarchical Group and Attribute-Based Access Control. Lastly, and most recently (2014), the work by Servos and Osborn [2014] attempts to create a formal general model of ABAC that provides a group-based hierarchical representation of object and user attributes. In this model, entitled Hierarchical Group and Attribute-Based Access Control (HGABAC), attributes are assigned both directly to access control entities and indirectly assigned through user and object attribute groups. Attribute groups help simplify administration of ABAC systems by allowing administrators to create user or object groups whose membership indirectly assigns sets of attribute/value pairs to its members. These groups are hierarchical and inherit attribute/value pairs from their parent groups allowing for more flexible policy representation when combined with the three-valued logic-based policy language proposed in the work.

The HGABAC policy language represents policies as C style boolean statements that may evaluate to TRUE, FALSE, or UNDEFINED. A resulting evaluation of TRUE implies that access should be granted, FALSE that it should be denied, and UNDEFINED if the policy can not be properly evaluated at the current time (equivalent to a result of FALSE for access control decision purposes). Policies are associated with a set of operations that they grant if satisfied. Example 2 presents a number of example policies that are possible in HGABAC.

Example 2. Possible policies supported by HGABAC:

—P_1 = (user.age $\geq$ 18 AND object.title = ''Adult Only Book'', read) Any user with an age of 18 or older can read the book with the title "Adult Only Book".
—P_2 = (user.id = object.author, write) A user can write to any object they are an author of.
—P_3 = (user.role IN {''doctor'', ''intern'', ''staff''} AND user.id != object.patient, read) A user can read a medical record if they have the role of doctor, intern or staff but only if they are not listed as a patient in that record.
—P_4 = (object.type = ''program'' AND object.required_certifications SUBSET user.certifications, run) A user can run a program if they have the required certifications listed in the programs required_certifications attribute.

Servos and Osborn [2014] show that their policy language and attribute groups are capable of emulating MAC, DAC, and hierarchical RBAC (though not separation of duties) and that their attribute groups result in less complex (in terms of the number of assignments and relations between access control entities) representations than standard (non-hierarchical) ABAC models under a number of hypothetical use cases.

5.1.2. Domain Specific Models.

While a handful of recent ABAC related works have sought to create "general" models, the more popular trend in modern access control literature has been the creation and formalization of "domain specific" ABAC models. A large focus has been given to the domains of cloud computing [Buehrer and Wang 2012], grid computing [Lang et al. 2006, 2009, 2010], web services [Yuan and Tong 2005; Shen and Hong 2006; Xia and Liu 2009; Shen 2009], and related areas including mobile computing [Covington and Sastry 2006] and cross-domain service-oriented architecture [Dan et al. 2012].

Cloud Computing. Buehrer and Wang [2012] propose an ABAC model based on class algebra, entitled CA-ABAC, intended to provide access control between federated educational clouds. CA-ABAC makes use of the non-probabilistic version of class algebra implemented by the Cadabia knowledge base [Buehrer et al. 2001] as a basis for its ABAC policy language. Example policies from Buehrer and Wang [2012] are given in Examples 3 and 4.

Example 3. Only students that have signed the contract/consent form (the form named okJim55) may read or execute course material owned by the teacher named Jim.

```
new Policy[n1] {
  agents: ``ENV.user in @School[A;B]
    .student{@Form[okJim55] in singedForms}'' ∧∧Query
  rights: @Rights[read,execute],
  objects: ``@Thing{owner = @Teacher[Jim]}
}
```

Example 4. Uses the environment's time attribute to block students from reading the answers to homework assignment 5 until after the due data.

```
new BlockPolicy[n5] {
  agents: ``ENV.user in @Student,'' ∧∧Query
  rights: @Rights[read],
  objects: ``@Homework[assignment5]
    {dueDate > ENV.date}.answer'' ∧∧Query
}
```

Buehrer and Wang [2012] outline a very informal description of their model, which mostly describes policy use and a hypothetical system architecture. While the prospect of using class algebra as a policy language may have potential, CA-ABAC lacks formalization or details on many of the key features of an ABAC system. No description is given of what constitutes an attribute in the model or their relation to users, objects, or the environment. Basing the policy language on Cadabia queries may lead to problems for real world use as the Cadabia open source project is no longer maintained.

Real-time Systems. Burmester et al. [2013] put forward the T-ABAC (real-Time Attribute-Based Access Control) model, that adds real-time attributes to the concept of rule-based ABAC to support highly dynamic real-time applications. Real-time attributes are defined as attributes whose value depends on time and is a member of an ordered set of availability labels that determines the "priority" of a subject's request, the "congestion" of a resource, or the "criticality" of the environment. Burmester et al. also provide a packet forwarding protocol that takes the priority of access requests into account and demonstrates the versatility of T-ABAC by discussing two possible

applications, a substation automation system, and a medical CPS. While the T-ABAC model does a good job of dealing with issues unique to real-time systems, it omits several core ABAC model components. No information is given about how policies are represented, evaluated, or applied to the model, and only the concept of real-time attributes is developed with regard to ABAC. As such, T-ABAC presents a sufficient basis for extending existing ABAC models to support real-time applications, but is missing necessary components to be a stand-alone model.

Collaborative Environments. Collaborative working and educational environments enable cooperative work, research, and learning through shared application or service resources. Collaborative applications and services include but are not limited to e-mail, wikis, instant messaging, group blogs, version control systems, courseware, and software to support shared document, workspace, task, and work flow management. As these applications have unique access control requirements, they have attracted a notable amount of attention in the access control literature, including a number of papers focusing on applying ABAC policies to collaborative systems. Such works include Smari et al.'s ongoing research project and multiple publications supporting ABAC for collaboration environments [Zhu and Smari 2008; Smari et al. 2009, 2014] and Liang et al.'s [2012] multiple-policy supported ABAC architecture for large-scale collaboration systems (MPABAC).

Smari et al. present an ABAC model aimed at collaboration environments [Zhu and Smari 2008] that incorporates trust and privacy into access control policies. They extend this model over a number of works [Smari et al. 2009, 2014] to fully formalize their notion of trust and privacy and illustrate their model with an implementation and detailed case study involving a multi-organizational collaborative crisis management system. Their model consists of a three-valued ("allow," "deny," and "NA") rule-based policy evaluation on subject and object attributes that integrates trust and privacy through special mutable trust and purpose attributes. Trust is considered to be "the degree that a subject will perform as expected in a certain given context" and is quantified as a real number between 0 and 1 and assigned as the value of a subject's trust level attribute. As a user performs requests upon the system, their previous behavior is assessed and used to determine if their future behavior deviates or conforms to what is expected (effecting the user's trust level). In addition to this dynamic notion of trust, a subject's trust level is also dependent on other subject attributes including the recommendation from others and the level of collaboration between organization of a requester and that of a resource. This trust level can then be included in access control policies to limit or expand a user's access to system resources based not only on traditional access policies but also their evaluated trust level. The concept of privacy is enforced by assigning a set of well-defined purposes to subjects and objects as an attribute that represents either for what purposes a subject may access an object or for what purposes an object may be accessed, respectively. Access to a specific object is allowed only if the purpose of the subject for accessing the object matches a purpose allowed by the object. While Smari et al.'s model successfully introduces trust and privacy to ABAC, it omits details on policy evaluation or a formalized policy language. Example policies are shown but no explanation is given for how the operations may work with the three-valued logic used by the model.

Liang et al. [2012] offer a model and architecture for MPABAC that addresses the access control issues inherent in large-scale device collaboration systems (i.e., mainly the large number of heterogeneous devices). Unlike other ABAC models, MPABAC models resources as devices (device attributes rather than object attributes, etc.) and focuses on limiting access to networked devices (e.g., seismographs, orchestrated lights) based on multiple policies possibly originating from different domains but evaluated

locally. The described architecture and implementation detail how XACML may be used to communicate access control information between different domains and enforce the MPABAC model. As MPABAC largely focuses on architecture and XACML use, the ABAC model itself omits details on how policies are evaluated or combined. Details on how attributes are represented (e.g., if they are sets, collections of values, or primitive data types) are similarly omitted and the notion of policies having a priority level is introduced but not fully formalized in terms of the MPABAC model.

Mobile Environments. Several efforts have advocated models of ABAC that are contextually aware of a user or resource's physical environment. Covington and Sastry's [2006] Contextual Attribute-Based Access Control (CABAC) investigates using the dynamic properties commonly available in a mobile environment (e.g., a user's current physical location) as attributes to support ABAC for mobile applications. Transaction attributes that are mutated or created based on a user's transactions with a service provider (e.g., a user may have an attribute that holds the total amount of money spent at a certain shop) are also supported as a special case of contextual user attribute. These attributes allow for access policies to be based around past transactions with a user. For example, a restaurant may have a policy that grants access to their WiFi connection to customers that have made a purchase in the last 24 hours. A custom authorization policy specification language consisting of constant symbols (e.g., object references), variable symbols (e.g., location and time), and operation symbols (e.g., +, -, /, *, AND, OR, $<$, $>$) is described but not formalized or demonstrated.

A similar work by Kerschbaum [2010] details an access control model for mobile physical objects that aims to apply access control to physical mobile resources embedded with RFID tags. Kerschbaum's model applies attribute-based visibility policies to supply chain information based on the contextual location of physical objects as they transverse multi-company supply chains. This is accomplished by extending Yuan and Tong's ABAC model for web services [Yuan and Tong 2005] (discussed later in this section) to include upstream and downstream visibility as an attribute for each pairing of subject and object to allow policies to be created based on an object's trajectory relative to a subject (i.e., whether a subject is upstream or downstream of an object's current location in the supply chain). Policy rules are specified using a Boolean function of the subject and resources attributes as shown in Example 5. In this example, a subject, *s*, may access the information pertaining to a resource, *r*, if the attributes "downstream" or "upstream" are in the attribute set produced by the pairing of *s* and *r*, that is, *ATTR(s, r)*. Such attribute sets are continuously updated based on the subject and resource's current physical location.

Example 5. Resource visibility policy:

```
access(s, r) ← ''downstream'' ∈ ATTR(s, r) ∨
               ''upstream'' ∈ ATTR(s, r)
```

A method for encoding such visibility policies in XACML is also described. XACML environment attributes are used in place of assigning attributes to pairings of subjects and resources (as XACML does not support direct assignment of attributes to subject resource pairs).

Grid Computing. Grid computing has been another common target of domain specific ABAC models as it presents unique access control requirements stemming from the distributed nature of grid computing, where resource providers and users may be in independent security domains. Lang et al.'s [2006, 2009] Attribute-Based Multipolicy Access Control (ABMAC) presents a model and Globus Toolkit release 4 (GT4) based

authorization framework for applying ABAC to grid computing. In addition to user, object, and environment attributes, ABMAC supports service and action attributes that allow attributes to be applied to grid services or a grid action, respectively. Policies differ from most rule-based ABAC models in that each policy is encapsulated and uses its own definitions and decision-making algorithms, allowing for independent evaluation without changing a policy's description. A similar but more informal work, Grid_ABAC [Lang et al. 2010], also uses GT4 to implement and demonstrate a grid-based ABAC model that supports action attributes and uses XACML as a policy language. Grid_ABAC, unlike ABMAC, largely focuses on being a grid authorization architecture and as such provides a more minimalistic ABAC model.

Web Services. By far the largest area of research in domain specific ABAC models is toward attribute and policy-based access control for web services. Identity-less access control such as ABAC provides a potential solution to furthering automated web service discovery and use by allowing access control decisions to be made without prior knowledge of the subject or their relation to the service provider. Of the many ABAC models targeting web services [Yuan and Tong 2005; Shen and Hong 2006; Dan et al. 2012; Xia and Liu 2009; Shen 2009; Zhang et al. 2014], most notable is the model by Yuan and Tong (ABAC for Web Services), upon which several other ABAC models [Kerschbaum 2010; Xia and Liu 2009] are based. Yuan and Tong describe ABAC in terms of authorization architecture and policy engineering and give an informal comparison between ABAC and traditional role-based models. Policy rules are defined as a Boolean function comparing the attributes of the subject making the request, the resource potentially being access, and the system's environment. If the function evaluates as true, access is granted to the subject, otherwise access is denied.

Two example policy rules from Yuan and Tong [2005] are shown in Example 6. Rule 1 (R_1) allows a subject, *s*, to access the ApprovePurchase web service resource, *r*, if they have a Role attribute with a value of "Manager." Rule 2 (R_2) allows any user access to a resource they own. That is, if their user ID is equal to the value of the ResourceOwner attribute for the given resource, *r*.

Example 6.

```
R1: can_access(s, r, e) ←
        (Role(s) = ‘‘Manager’’) ∧
        (Name(r) = ‘‘ApprovePurchase’’)
R2: can_access(s, r, e) ←
        (UserID(s) = ResourceOwner(r))
```

While Yuan and Tong's model is limited, only giving an overview of subject, object, and environment attributes and their relation to policies, it was an earlier effort that served as the basis for more formalized future works. In addition to the model, an authorization architecture is introduced that uses XACML to securely communicate attributes, policies, and access control decisions between a number of actors.

Shen and Hong [2006] propose WS-ABAC, a more extensive but still relatively simplistic ABAC model designed for web services accompanied by an XACML-based authorization architecture. In the WS-ABAC model, policies are based on a straightforward tuple language that is mapped to XACML when used in their authorization architecture. Attribute constraints are expressed as a series of attribute conditions, <*Attribtue Name*> <*Operation*> <*Value*> statements, combined with logical AND (represented as $\cap$) or OR (represented as $\cup$) operators. Valid attribute condition operations are limited to $>, <, \geq, \leq, =, \neq$. In Example 7, constraint C_1 limits access to a web service to a manager who is accessing the service between the hours of 9:00 AM

and 5:00 PM from the office. Constraint C_2 limits access to a clerk when the system load is low or to a manager at any time or system load.

Example 7. Example WS-ABAC Attribute Constraints

C_1: Identity=‘‘manager’’ ∩ Time≥9:00 ∩ Time≤17:00 ∩ Location=‘‘office’’
C_2: Identity=‘‘clerk’’ ∩ System_load=‘‘low’’ ∪ Identity=‘‘manager’’

WS-ABAC policies are defined as the triple $<S, srv, C>$, where S is the set of subjects to which the policy pertains, *srv* is the service the policy grants access to, and C is the attribute constraint. Access to a service is only granted if (1) there exists a policy triple containing the requested service, (2) the user, U, making the request is a member of S ($U \in S$), and (3) the attribute constraint, C, evaluates to true. As with Yuan and Tong's ABAC for Web Services, this work presents a minimalistic model and mostly focuses on an architecture that uses XACML and attribute-based policies to provide authentication for web services (as opposed to a complete and/or foundational model of ABAC).

A number of later publications have followed in the same suit, providing minimally sufficient models with accompanying XACML-based architectures targeting web services or service-oriented architectures (SOA). Dan et al. [2012] create and implement an XACML architecture for cross-domain SOAs. Xia and Liu [2009] study using action and attribute-based models for web services and develop a limited ABAC model and XACML architecture that extends the work of Yuan and Tong. Shen [2009] presents SABAC, an informal semantic-aware ABAC model for web services that makes use of present standards, including XACML. Finally, Zhang et al. [2014] describe a largely informal ABAC security model for service-oriented computing that adds the notion of trust as well as offering an authorization architecture for web services based on combining existing works (mainly SAML and XACML). However, few details are provided on their ABAC or trust model, as more attention is given to the authorization architecture.

Digital Libraries. An earlier work by Adam et al. [2002] identified the need for attributes to deal with the challenging requirements of providing access control for digital libraries. Digital libraries are information systems that facilitate the storage, retrieval, and acquisition of knowledge between creators, consumers, and librarians on a global scale. Adam et al. suggest a novel access control system for protecting the Global Legal Information Network (GLIN), a digital library created by the Law Library of Congress for making laws and legal decisions accessible to citizens, legislators, government, and private sector officials.[1] Their model grants privileges based on user credentials (sets of typed attributes relating to the same topic or structure, e.g., an *employee* credential may contain an *age*, *address,* and *salary* attribute) and object concepts (conceptual hierarchies extracted from the content of an object using a document management mechanism built into GLIN [Holowczak 1997]). Both credentials and concepts are hierarchical. Credentials types (declarations of what attributes are contained in a credential, their type and possible values) are organized into a hierarchy such that a credential type inherits all attributes of the credential type proceeding it in the hierarchy. For example, if an *employee* credential type specified that it contains the attributes *age*, *address,* and *salary* and an *international_employee* credential type specified that it contains the attributes *nationality* and *visa*, it would also gain the attributes *age*, *address,* and *salary* if *international_employee* was a child of *employee* in the credential type hierarchy.

[1]http://www.glinf.org.

A simple credential constraint specification language is introduced that allows for the evaluation of user's attribute values (or their assignment to a specific credential) using rudimentary operations ($=, \neq, <, >, \leq, \geq, \subset, \subseteq, \supset, \supseteq, \not\subset, \not\subseteq, \not\supset, \not\supseteq, \in, \notin$). Constraint expressions take the form of $X.a\ OP\ v$, where X is a variable representing any user in the system, a is an attribute name, OP is an operation, and v is a value (for example, $X.age > 18$ would specify all users with an age over 18). Constraint expressions can also simply be a credential type to specify all users assigned to a given credential (including children of that credential in the credential type hierarchy). For example, the expression $employee(X)$ would specify all users who are employees. These constraint expressions are used in Access Authorizations to create the access policies of the system. Access Authorizations are tuples consisting of a credential specification (one or more credential expressions joined with AND or OR symbols), entity specification (denotes the concepts, objects, or parts of objects the authorization refers), privilege (a valid operation on an object), and sign (whether the authorization is positive, grants the privilege, or negative, forbids it). Example Access Authorizations are shown in Example 8.

Example 8. Some Possible Access Authorizations:

— A_1 = (employee(X), 2016 Income Report, view-all, +) Allows all employees to view the "2016 Income Report".
— A_2 = (international_employee(X) $\wedge$ X.nationality = Canadian, 2016 Income Report.Canada part, update, -) Forbids international employees from Canada from updating the Canada part of the "2016 Income Report".
— A_3 = (X.age $\geq$ 18, Book of Guns $\wedge$ Book of Drugs, view-all, +) Allows any user with an age of 18 or over to view the "Book of Guns" and the "Book of Drugs".

Adam et al. also [2002] provide details about a supporting system architecture and protocol, discuss an implementation of their model, and explain how administrative operations are performed. Later work by the same authors [Ferrari et al. 2002] introduces an authorization system for digital libraries that utilizes this access control and authorization model.

5.2. Hybrid Models

Hybrid models of ABAC aim to combine attributes into existing models of access control or to extend the traditional models with identity-less or policy-based access control concepts. This includes both early attempts at adding parameterized roles and permissions to RBAC as well as more modern efforts to unify ABAC with alternative access control models such as relationship-based access control (ReBAC) and behavior-based access control (BBAC). Kuhn et al. [2010] describe a number of hypothetical strategies for adding attributes to RBAC:

Dynamic Roles. Roles are assigned dynamically based on the user's and environment's attributes, providing identity-less access control for RBAC-based systems. Most dynamic role-based hybrid models lack object attributes or a means to dynamically assign permissions to roles, and as such, lack the flexibility of ABAC to limit access based on the content of objects (e.g., only allow users to view medical records in which they are the patient). This leads to what has been described as an "explosion" [Jin et al. 2012b] of role-permission assignments or the creation of a large number of private roles.

Attribute-Centric. Roles are considered to be just another attribute of a user. No role-permission relation is created and permissions are assigned through policies. If no special consideration for roles is provided in an Attribute-Centric model this could be seen simply as "pure" ABAC modeling RBAC. As this can be seen as

equivalent to "pure" ABAC in most cases, it is deprived of the advantages of RBAC (simple administration, auditability, straightforward separation of duties, etc.).

Role-Centric. The maximum permission set available in a given session is constrained by attribute-based rules. Constraint rules are used only to reduce permissions available to the user and never expand them (differentiating it from role parameterization). Few details are given about how this strategy may be implemented or if it is different enough from existing models of parameterized RBAC to warrant its own strategy. To date, only one published work is known to specifically utilize this strategy [Jin et al. 2012b].

In addition to the strategies described by Kuhn et al., role parameterization [Ge and Osborn 2004; Giuri and Iglio 1997; Abdallah and Khayat 2005] can be seen as a viable option for ABAC-RBAC hybridization. In Parameterized RBAC, permissions (and in some cases roles) are parameterized with conditions that must be met before access is granted to a subject. Often these conditions involve attributes of the object being accessed but may also include attributes of the user and environment (e.g., time).

We categorize the ABAC Hybrids reviewed in this section into the following subcategories:

Parameterized Role-Based Access Control. RBAC models based around extending RBAC by parameterizing permissions and/or roles as described in Section 5.2.1.

Attribute-Based Role Assignment. Models that extend RBAC to add attributes as described in Kuhn et al.'s Dynamic Roles strategy (i.e., assigning roles via user attributes). These models are reviewed in Section 5.2.2.

Attribute-Centric. Models that extend RBAC to add attributes as described in Kuhn et al.'s Attribute-Centric strategy that would not be classified as "pure" models of ABAC. These models are reviewed in Section 5.2.3.

Role-Centric. Models that extend RBAC to add permission filtering based on attributes as described in Kuhn et al.'s Role-Centric strategy. To date, only Jin et al.'s [2012b] Role-Centric Attribute-Based Access Control (RABAC) is known to exist in this category. This model is reviewed in Section 5.2.4.

Unified Models of Access Control. Access control models that combine ABAC with with alternative access control models (i.e., non-traditional models) as described in Section 5.2.5.

Table VI in the Appendix summarizes and compares the most notable of these hybrid models using similar criteria to the comparison between "pure" ABAC models found in Appendix Tables III and IV (criteria defined in Table II).

5.2.1. Parameterized Role-Based Access Control. Parameterized Role-Based Access Control (sometimes abbreviated PRBAC [Abdallah and Khayat 2005]) can be seen as an early first step toward ABAC. In PRBAC, permissions normally modeled as object, access mode pairs in RBAC are parameterized with a condition that must be met before the permission is granted to a subject. In Giuri and Iglio's Role Template model [Giuri and Iglio 1997], RBAC permissions are extended with a logical expression referred to as the privilege restriction.

This restriction is evaluated against both the object on which access is requested and the returned value of predefined functions. One example (given by Abdallah and Khayat [2005]) would be if permissions included *"(delete, PatientRecord, PatientRecord.State = 'discharged'),"* then the delete operation would be permitted on any patient record that is in a "discharged" state; similarly, the permission*"(delete, PatientRecord, today() in*

[Mon..Fri])" would permit the delete operation only on week days (Monday to Friday). Additionally, role templates are defined that extend the concept of roles to "encapsulate and compose parameterized privileges." These templates act as a function that takes a set of values (related to the object the role grants access to) and returns a set of parameterized permissions that make up a role. For example, the role template (also taken from [Abdallah and Khayat 2005]) given in Example 9 would produce the template instance given in Example 10 if the values *prj = "PRJ1"* and *sal = 1000* are used.

Example 9. Example role template.

```
R<prj, sal>= role(
  (select, Employee, Employee.project = prj),
  (update, Employee, Employee.project = prj ^ Employee.salary <sal))
```

Example 10. Resulting role instance for values prk = "PRJ1" and sal = 1,000.

```
R,<''PRJ1'', 1000>= role(
  (select, Employee, Employee.project = ''PRJ1''),
  (update, Employee, Employee.project = ''PRJ1'' ^ Employee.salary <1000))
```

While the role templates and parameterized permissions described by Giuri and Iglio [1997] may provide some advantages over classical RBAC, they do not consider the attributes of the subject, limiting their privilege restrictions to only attributes of objects. This makes policies such as "each student can access their own transcript" difficult to implement without assigning a unique template instance to each student.

The work by Ge and Osborn [2004] toward parameterized roles to support XML databases provides a PRBAC solution that includes both the attributes of subjects and the contents of objects. Ge and Osborn extend the role graph model [Nyanchama and Osborn 1999] to parameterize privileges with XPath-like [Clark and DeRose 1999] logical expressions that contain variables determined at runtime based on attributes defined in a user's session. In an example given in the paper, the parameterized privilege pair *"(//Student[@StudID = param1]//GeneralInfo, update)"* would grant access to update a student's record general info section if the user's student id attribute matched the student id in the record. Roles are adapted to support parameterized privileges and the implications of parameterization on inheritance in the role graph is considered. While this extension supports object attributes (limited to the contents of the object), it is only applicable to the narrow domain of restricting access to XML-based databases as opposed to a generic access control solution.

A number of similar PRBAC works have attempted to add logical expression-based policies to RBAC permissions, including Abdallah and Khayat's [2005] PFRBAC (an extension of FRBAC [Khayat and Abdallah 2003]) and Lupu and Sloman's [1997] model for reconciling Role Based Management (RBM) and RBAC. Although these and other PRBAC works add aspects of policy- and attribute-based access control to RBAC, they fail to provide the identity-less nature of modern ABAC systems. Users (or subjects) still require assignment to roles (in most cases done manually), requiring pre-existing knowledge of both the user and their place in the organization. While sufficient for conventional access control scenarios, identity-based access control like PRBAC fails to provide the flexibility required for emerging computing paradigms, including service-oriented architectures (e.g., web services) or dynamic environments as commonly found in cloud computing.

5.2.2. Attribute-Based Role Assignment. Models based on Attribute-Based Role Assignment or "Dynamic Roles" as defined by Kuhn et al. [2010] allocate roles to subjects based on the attributes of the subject and environment at runtime. In most cases,

administrator created policies are defined via policy languages that relate attributes to constant values (e.g., checking if a user's age is greater than 18) and role assignment is performed when a subject first creates a session with the system based on the outcome of these policies, the user's attributes (e.g., age), and the current state of the environment (e.g., current day of the week). These roles may be limited to a set of possible roles assigned to the user (identity-based) or made totally dependent on attributes with no pre-existing knowledge of the user (identity-less).

Al-Kahtani and Sandhu [2002] introduce identity-less access control concepts into RBAC in their Rule-Based RBAC (RB-RBAC) model by automating the assignment of roles at runtime based on a user's attributes. In RB-RBAC, rules defined in a custom policy language determine the set of roles a user is assigned based on attributes provided with the user's credentials. Policy rules take the form of *Attribute_Expression* $\rightarrow$ *Roles* statements where *Attribute_Expression* is a Boolean statement involving attribute names/values and *Roles* is one or more roles granted if the user's attributes satisfy the attribute expression.[2] Example 11 demonstrates three rules that are possible in their policy language. Rule 1 (R_1) grants the *Guest* role to any user between the hours of 9 AM and 5 PM. In rule 2 (R_2), users from Japan or New Zealand who are also 20 years or older are granted the *Adult* role. Finally, in rule 3 (R_3), users from Canada, the USA, or Mexico who are 18 years or older are granted both the *Adult* role as well as the *North American* role.

Example 11.

R_1: (Time IN (900 .. 1700)) $\rightarrow$ Guest
R_2: (Age $\geq$ 20) AND (Country IN {Japan, New Zealand}) $\rightarrow$ Adult
R_3: (Age $\geq$ 18) AND (Country IN {Canada, USA, Mexico})
$\rightarrow$ Adult AND North American

Seniority levels are used to denote an attribute's value dominating another value in cases where the order of values is not clear (e.g., strings or sets rather than numerical values), allowing operations such as less than ($<$) or greater than ($>$) to be performed on values of any type. The versatility of the model is demonstrated through a number of real life cases; however, the lack of object attributes limits the flexibility of possible policies compared to those possible in most "pure" ABAC models.

A number of approaches [Jin and Fang-chun 2006; Cirio et al. 2007; Cruz et al. 2008, 2009; He et al. 2011] have attempted to use Semantic Web Technologies, such as Web Ontology Language (OWL) [McGuinness et al. 2004], Semantic Web Rule Language (SWRL) [Horrocks et al. 2004], and SPARQL Protocol and RDF Query Language (SPARQL) [PrudHommeaux and Seaborne 2008], to both model hierarchical RBAC and extend it with attribute-based dynamic role assignment. Cirio et al. [2007] propose both a hybrid RBAC-ABAC model and a supporting framework based on OWL Description Logic (OWL-DL) in which attributes are used to classify subjects into access control roles. While all basic RBAC elements are formalized into an OWL-DL ontology and details for expanding the expressiveness of OWL-DL with SPARQL are given, Cirio et al. [2007] do not fully model the attribute-based aspects of their ontology. Details on how attributes are defined, assigned, related to users, or how they may be combined with their framework are not provided. Cruz et al. [2008, 2009] describe a "Constraint and

[2]The grammar of the policy language presented in Al-Kahtani and Sandhu [2002] allows for more flexible policy rules that include more complex constrains, restrictions, and role combinations. However, the article leaves most of these to future work/extensions.

Attribute Based Security Framework for Dynamic Role Assignment" focused partly on using a user's physical location for role assignment. In this approach, role membership can consist of both previously know directly assigned users as well as users dynamically assigned based on the content of their attributes and the constraints placed on the role. Rather than employing a policy language like most ABAC works, constraints are defined as attribute name, constraint pairs (e.g., $<Age, \geq 18>$) that are assigned directly to roles to limit their assignment. Semantics for role inheritance and constraint dominance are given in addition to a description of an OWL-DL ontology-based prototype. Finally, both He et al. [2011] and Jin and Fang-chun [2006] have also produced semantic web-based RBAC models that add elements of ABAC. Both works represent and provide a means to reason about hierarchical RBAC in description logic, He et al. using SWRL [Horrocks et al. 2004] and Jin and Fang-chun using ALC(D) [Baader and Hanschke 1991]. Both also use attribute-based policies for role assignment. The main difference between these models is the limitations put on attributes and support for separation of duties; He et al. limit attributes to user credentials that have been verified by a trusted third party (a process described in their accompanying architecture) and support classical RBAC separation of duties, while Jin and Fang-chun allow temporal attributes in addition to the attributes of subjects but lack any notation of separation of duty style constraints.

Shafiq et al. [2005] propose an agent-based framework for attribute-enhanced RBAC in distributed environments that extends the Generalized Temporal Role-Based Access Control (GTRBAC) model [Joshi et al. 2005]. In this framework, users are both directly assigned roles before hand and allowed to request additional roles at runtime based on their self-declared attributes and the amount of trust a service provider has in those attributes (determined partly based on additional credentials submitted by the user). In addition to allowing temporal constraints on activating roles (e.g., only allowing the role employee to be activated between 9AM and 5PM), the framework also allows constraints to be placed on the duration a role can be enabled in a given time interval, defined either for a single session or a total duration of all sessions in which the role is active. The X-GTRBAC [Bhatti et al. 2005] XML-based policy language is extended to support SAML-based assertions and attribute-based authorizations used in the framework. While this work presents a novel extension to GTRBAC to support hybrid ABAC in cases where a single trusted attribute authority may not be available, like RB-RBAC it also omits support for object attributes, limiting the expressiveness of possible policies.

A number of comparable models aims to provide analogous support for attribute-based role assignment for web services and service-oriented environments, including the work done by Zhu et al. [2008] and Wei et al. [2010]. Zhu et al. put forward their "general attribute based role-based access control" (GARBAC) model aimed at web services while Wei et al. introduce their "Attribute and Role Based Access Control" (ARBAC) model aimed at service-oriented environments. Both models provide hybrid ABAC for service-oriented architectures and support a similar set of features including object attributes and hierarchical roles. While these models add object attributes (something lacking in other models in this subcategory), they lack formal definitions of the policy language being used or the semantics behind it. It is also left unclear how object attributes might be used in role assignment policies in practice if assignment takes place before requests on specific objects are performed (in GARBAC, a constraint on role-permission assignment is hinted at but only shown partly in an example case study).

5.2.3. Attribute-Centric. Models based on the "Attribute-Centric" strategy, as defined by Kuhn et al. [2010], have the characteristic of incorporating attributes into RBAC model roles as just another attribute of the user and not necessarily a separate access control entity onto which permissions are assigned. Instead, permissions are assigned based on evaluating policies relating attributes of users, the environment, objects, etc., with each other and constant values. If no special consideration for roles is given, this is equivalent to the "pure" ABAC models described in Section 5.1 that can be seen as a more generalized access control model than RBAC (as it is possible to emulate RBAC configurations in ABAC policies). If special consideration for roles is provided, such as using role-based separation of duties, the model is considered to be an ABAC-RBAC hybrid and described in this section.

The most notable Attribute-Centric work that does not fall into the category of "pure" ABAC is Huang et al.'s [2012] "a framework integrating attribute-based policies into role-based access control" that models RBAC on two levels. A front end (or "aboveground") level presents itself as a traditional RBAC model extended only with environmental attributes (applied to both user-role and role-permission assignments) and a back end (or "underground") level emulates the simplistic RBAC front end using attribute-based policies. This departmentalizing allows routine access control operations and auditing/review to be performed on the simpler RBAC front end, while still allowing the more complex administration and fine grained attribute-based policies to be created in the ABAC back end.

Underground-level policies are divided into two categories: Role-permission assignment policies that determine assignment of permissions to roles, and user-role assignment policies that determine the assignment of users to roles. Both types of policies are specified using first-order logic (FOL) expressions that follow structures shown in the following:

Role-Permission Assignment Policy Structure

```
rule_id {
  target {
    role_pattern;
    permission_pattern {
      operator_pattern;
      object_pattern;
    }
    environment_pattern;
  }
  condition;
  decision.
}
```

User-Role Assignment Policy Structure

```
rule_id {
  target {
    user_pattern;
    role_pattern;
    environment_pattern;
  }
  condition;
  decision.
}
```

Where *patterns* are FOL expressions that define a set of environmental states, set of roles, set of users, set of object, etc., as appropriate and comprise the *target* of the rule (the access control entities to which this rule applies). The *condition* is an FOL expression that defines conditions that must be met for the role or permission to be assigned and the *decision* defines the exact role or permissions assignment that will be made. Example 12 shows a user-role assignment policy that grants any role of type "employee" to any user (as no user pattern is given) located in London. The granted roles are only valid in environments matching the environmental pattern specified. In this case, only on weekdays and while the system mode is set to "normal."

Example 12.

```
rule: {
 target: {
  role_pattern(r): r.type = ‘‘employee’’;
  environment_pattern(e): {
    Time = ‘‘Weekday’’
    and Mode = ‘‘Normal’’}
 }
 condition: {
  u.location = ‘‘London’’;
 }
 decision: add (u,r,e) in URAe.
}
```

While this dual level model simplifies administrating a large scale ABAC system, this benefit is only maintained if policies of the back-end ABAC model conform to those reviewable in a standard RBAC framework. Back end policies that grant roles based on non-identity related attributes (e.g., location, time) rather than limit activation of or put constraints on previously assigned roles can easily lead to issues when attempting to determine the set of users who have access to a given role or permission (as is the case with most ABAC systems). This forces the role/policy engineer to choose between creating an identity-less access control system or one that is easily auditable.

5.2.4. Role-Centric. Jin et al.'s [2012b] role-centric attribute-based access control (RABAC) extends the NIST RBAC model [Ferraiolo et al. 2001] to create the first attempt at a formal Role-Centric RBAC-ABAC hybrid model. RABAC follows Kuhn et al.'s [2010] approach of reducing the number of permissions available to a subject in a traditional RBAC session based upon the current value of attributes (in this case only user and object attributes). Permission filtering policies, defined in a custom Common Policy Language (CPL) [Jin et al. 2012a] based language, are used to reduce the maximum permission set in a given session by checking each permission against all applicable filtering policies. The applicability of each policy is determined by a secondary "condition" policy assigned to each filtering policy that determines if it should be applied to a given permission based on the attributes of the object. This method is used to constrain permissions without significantly modifying the NIST RBAC model (only the set of permissions available to a subject in a given session are effected) enabling other concepts, such as separation of duties or the role hierarchy from the NIST model, to be directly applied to RABAC without modification.

While this work does provide a first attempt at a role-centric model, it is unclear if it poses a significant benefit over preexisting models of PRBAC. Both offer an identity-based solution that constrains role-permission assignment, the main difference being that PRBAC changes the process of the role-permission relation such that permission assignment is determined at runtime while RABAC keeps the relation unchanged and filters permissions out during session creation. Jin et al. [2012b] argue that this difference enables RABAC to make use of the NIST RBAC administrative model while PRBAC models would require new and more complex administration models.

5.2.5. Unified Models of Access Control. We define *Unified Models of Access Control* as any models of access control that attempt to combine two or more non-traditional models of

access control into a single unified model. For the purposes of this section, only models that include ABAC are considered. Cheng et al. attempt to combine relationship-based access control (ReBAC) with ABAC in their UURAC$_A$ model [Cheng et al. 2014] by extending the user-to-user relationship-based access control (UURAC) [Cheng et al. 2012] model. ReBAC-based models provide access control for Social Network Systems (SNS) based on a subject's relations with other users and entities in the social network. For example, a user may create an access policy to limit access to viewing their profile to only friends or friends of friends (i.e., limiting access to the profile to users with a user-to-user relationship depth of 1 or 2 from the profile owner on the social graph). Cheng et al.'s UURAC$_A$ adds attributes to both the nodes (users and resources) and edges (relationships) of the social graph, representing attributes of users, resources, and relations (type, weight, trust, etc.). A custom policy language (based on the language from UURAC) enables users to restrict access to owned resources based on a combination of attributes and relations. The following example policies (taken from Cheng et al. [2014]) restrict access to a profile based on users who share at least five common friends who are students (P_1), restrict access to a profile to friends in common with "Bob" (P_2), and restrict access to a photo to users who are within 3 hops of the owner on the social graph with a minimum trust value of 0.5 at each hop (P_3).

Example 13.

P_1: $\langle profile\_access, (u_a, ((ff, 2) : \exists[+1, -1], occupation(u) = \text{“}student\text{”}, count \geq 5))\rangle$
P_2: $\langle profile\_access, (u_a, ((ff, 2) : \exists[+1, 1], name(u) = \text{“}Bob\text{”}, \_))\rangle$
P_3: $\langle read, Photo1, (u_a, ((f*, 3) : \forall[+1, 1], trust(r) \geq 0.5, \_))\rangle$

While UURAC$_A$ successfully adds attributes to UURAC, there are some possible privacy concerns resulting from allowing end users to define their own attribute-based policies (something that is not unique to UURAC$_A$ but any ABAC model that allows users to create policies to protect their own resources/objects). For example, if a user, Alice, has a private profile on an SNS and an attacker, Eve, wishes to obtain some private information from that profile that is also an attribute describing Alice (e.g., location, age, gender, occupation). Eve could generate a large number of resources that would be appealing to Alice to view (e.g., a link to a picture with the text "Is this you in this picture?") and protect each resource with a policy that contains a guess at the value of one of Alice's attributes (e.g., ($name(u) =$ "*Alice*") $\wedge$ ($age(u) = 18$), ($name(u) =$ "*Alice*") $\wedge$ ($age(u) = 19$), ($name(u) =$ "*Alice*") $\wedge$ ($age(u) = 20$)). Alice would only be able to access the resource with the correct value and Eve would be able to determine this value by checking which resource is accessed. For example, if the set of resources were posts containing a link, each to a different image on Eve's website, Eve could determine the value of the attribute by matching the accessed image to the policy used to protect the accessed resource. This sort of attack could also be conducted more efficiently by using ranges of values for the attributes Eve is guessing at (e.g., ($name(u) =$ "*Alice*") $\wedge$ (($age(u) > 10$) $\vee$ ($age(u) < 20$))) to narrow down the value with fewer resources generated.

Che et al.'s [2010] "Behaviours and Attributes Based Access Control" (BABAC) attempts to unify behavior-based access control (BBAC) and ABAC to provide a novel access control solution for network virtualization. In BABAC, user behaviors (a single or sequence of actions performed by a user) are quantized and divided into three categories; Time-Lasting Behavior (a single persistent action that last for a fixed amount of time), Instant Behavior (a single action that happens instantly and has no associated

length of time), and Multi-Action Behavior (a combination or sequence of Time-Lasting and Instant behaviors). These behaviors are then used in combination with user and environment attributes to define access control policies that restrain access to resources both before and after permissions are assigned (e.g., a user's access to a resource could be revoked if they spend too much time performing a single action). The BABAC revocation policy in Example 14 (from Che et al. [2010]) revokes read access to the resource "FinancialPlan" if the user views the resource for more than 60 minutes, attempts to perform an illegal copy operation, or more than three users are trying to access this resource at one time. The time-lasting behaviors (TB), instant behaviors (IB), and multi-action behaviors (MB) that will be used in the policy are specified before the revoke policy expression.

Example 14.

```
Resource = ''FinancialPlan''
Action = ''Read''
TB = ''TotalViewTime''
IB = ''PerformIllegalCopy''
MB = ''TotalSeveringUser''
Revoke(U,R,A) ⇐ { TotalViewTime(U) ≥ 60 minutes
  ∨ PerformIllegalCopy(U) = true
  ∨ TotalSeveringUser ≥ 3 users }
```

To support access requests between independent virtual networks, user attributes are divided into three types; Global Attributes (user attributes obtained from a virtual network independent global attribute authority trusted by all virtual networks), Intra-domain Attributes (user attributes defined locally by an individual virtual network that access is currently being requested upon), and Trust-domain Attributes (user attributes imported from remote virtual networks that are trusted by the current network upon which access is currently being requested). Example 15 shows how these attributes may be used in a BABAC policy to grant access to a resource (the same financial plan as in Example 14). In this case, a user is allowed read access if they have a global security level of 5 or greater, have a job title of "junior-manager" in the local network, or have a job title of "senior-manager" in a trusted network and are not located in department C of a trusted network.

Example 15.

```
Resource = ''FinancialPlan''
Action = ''Read''
GAttr = ''SecureLevel''
IAttr = ''JobTitle'', ''Location''
TAttr = ''JobTitle'', ''Location''
Allow(U,R,A) ⇐ { SecureLevel(U) ≥ 5
∧ (JobTitle(U) ≥ IAttr(junior-manager) ∨ JobTitle(U) ≥ TAttr(senior-manager))
∧ Location(U) ≠ TAttr(dept.C) }
```

One last notable effort, is Han et al's [2009] work toward a united access control model that combines ABAC, RBAC, and task-based authentication control (TBAC). In Han et al.'s united model, TBAC is extended with attribute-based constraints (limited to user and object attributes) in addition to hierarchical role-based assignment of task permissions. Permissions are divided into Executing (permission to execute

a task), Supervising (permission to initiate, approve, dispense, or administrate task execution), and Invoking (permission to initiate a task request and acquire the result) permissions that are granted by roles. ABAC is used largely for negotiating identity-less role assignment with external users and functions similarly to attribute-based role assignments.

While unified models provide interesting new takes on existing non-traditional models, they are often limited in their applicability to real-world access control scenarios, instead targeting niche access control scenarios or domains. UURAC$_A$'s application is limited to SNS, BABAC to network virtualization, and Han et al.'s united model to systems in collaborative commerce. Additionally, combining models often leads to increased complexity such as is the case in Han et al.'s united model where administrators are required to deal with attributes, policies, role assignments, role hierarchies, workflows and tasks for both internal and external users; all in a single access control system. While this provides a large number of fine grained configuration points, it's questionable how manageable or auditable real-world implementations would be, especially in systems with a large number of access control entities.

6. OPEN PROBLEMS

As ABAC research is still largely in its infancy, the list of open problems related to ABAC systems and implementations is extensive. The majority of these problems stem from the increased complexity attribute, and policy-based access control introduces for the sake of increasing the flexibility and generality of access control policies. While hybrid ABAC models and frameworks aim to remedy these issues by extending proven traditional models, this is often done at the cost of flexibility or removing the identity-less nature of ABAC. This section outlines the most common problems identified and discussed in the recent literature (namely the works reviewed in Section 5) relevant to ABAC and to a lesser extent, policy-based access control in general.

6.1. Foundational Models

One frequently discussed issue [Jin et al. 2012a; Hu et al. 2013; Servos and Osborn 2014] is the lack of an agreed upon reference and/or foundational model of ABAC. While a large number of ABAC models have been published, they have predominantly been domain specific and limited to a particular use case (e.g., web services) or hybrid models that lack the versatility of "pure" models. Of the generalized models discussed in Section 5.1, only three [Jin et al. 2012a; Servos and Osborn 2014; Zhang et al. 2005] are both formal and complete models, none of which have garnered mainstream acceptance as "the standard" model of ABAC.

To date, the most frequent works cited as "the model of ABAC" have been XACML, Wang et al.'s [2004] logic-based framework for ABAC, and Yuan and Tong's [2005] ABAC for web services. However, these works are problematic as foundational models for a number of reasons. As XACML is simply an access control policy language, it lacks any kind of formal model of ABAC despite its support for attributes, making it at best only one component of a larger model. Wang et al.'s logic-based framework, provides a start toward a generic foundational model but mostly concentrates on modeling policies and their evaluation and can not be seen as a complete model of ABAC. Yuan and Tong's ABAC model for web services, while an early effort and the basis for several other models [Kerschbaum 2010; Xia and Liu 2009], is simplistic and specific to a limited domain. Perhaps the most promising, but yet to be completed or published, work is the purported effort at NIST toward a formalized family of ABAC models. During the NIST Attribute Based Access Control Workshop held on July 17, 2013, limited details

on the "Framework of ABAC models" were presented by David Ferraiolo that defined four families of ABAC models; $ABAC_{rule}$, $ABAC_{rule\text{-}hier}$, $ABAC_{rel}$, and $ABAC_{rel\text{-}history}$. Unfortunately, to date, few details and no formal definitions are available for these models (the only source being an unrefereed set of presentation slides [Ferraiolo 2013]).

Beyond model adoption or creation by a standards organization, a possible solution may lie in the suggestion of Barker [2009] for access control research to avoid "developing the next 700 particular instances of access control models" and instead focus on unifying meta-models. A meta-model of ABAC, or perhaps all policy-based access control in general, could provide a unified model for describing and reasoning about ABAC without necessitating the need for creation of new models for each small extension of the concept.

6.2. Emulating and Representing Traditional Models

It has been claimed that ABAC is a more general model of access control as it is capable of emulating the traditional models [Chadwick et al. 2003; Jin et al. 2012a; Lang et al. 2009; Servos and Osborn 2014; Park and Sandhu 2004]; however, as of now, this has only been demonstrated in the literature in a largely informal and shallow manner. The work by Jin et al. [2012a] has presented the most formal effort to date, demonstrating how $ABAC_{\alpha}$ can be constrained to model DAC, MAC, and hierarchical RBAC. However, only a single possible representation is given for each classical model (a number of which assume a partially ordered set may be used as an attribute's value) and the separation of duty constraints of RBAC are not modeled. A deeper exportation and evaluation of the different possible methods of representation are required to both develop best practices for aiding in the transition to ABAC (e.g., converting existing traditional systems to ABAC systems) and formally proving that ABAC can model all possible DAC-, MAC-, and RBAC-based policies.

6.3. Hierarchical ABAC

In hierarchical RBAC, the role hierarchy allows for roles to be related in a way that more closely resembles that of actual organizations. This allows for more simplistic administration, both in terms of role engineering and reviewability of existing role-based policies. Most "pure" models of ABAC, however, lack this type of inheritance and expressiveness. While a role can be easily modeled as a single attribute of a subject, this simplistic representation is unable to emulate the hierarchical nature of RBAC without allowing for complex data types in an attribute's value (as is done in Jin et al. [2012a] $ABAC_{\alpha}$) or unmaintainably complex policies. A more simplistic means of providing hierarchical administration is required for "pure" ABAC to be competitive with RBAC and hybrid models.

A possible solution may be found in "attribute user groups" [Servos and Osborn 2014], hierarchical groups that inherit sets of attributes from their parent groups and allocate these attributes to their members (similar to how roles in hierarchical RBAC could be seen as allocating permissions to the role's membership). This technique could also be applied to objects and other access control entities onto which attributes may be assigned. Another approach is to allow attributes to have inheritance relations directly with other attributes, such that a child attribute supersedes the parent attribute in policies. For example, if both the attributes "cs_faculty" and "cs_graduate_student" are children of the attribute "cs_department," being assigned "cs_faculty" or "cs_graduate_student" would fulfil a policy requiring a user to be assigned the "cs_department" attribute. This is similar to the attribute hierarchies

described in Wang et al.'s [2004] ABAC framework as well as other models, but potentially limits the usefulness of ABAC, as attributes no longer have values (instead, each attribute hierarchy could be seen as a single attribute with members being the possible values for the attribute).

6.4. Auditability

An important aspect of access control for both legal and security reasons is the ability to easily determine the set of users who have access to a given resource or the set of resources a given user may have access to (sometimes referred to as a "before the fact audit"). In RBAC, this is relatively straightforward, normally just requiring the system to calculate the union of the set of effective privileges from each role the user is assigned. However, in ABAC this is considerably more complicated [Hu et al. 2013]. As ABAC is an identity-less access control system and users may not be known before access control request are made, it is often not possible to compute the set of users that may have access to a given resource. Even in cases where the identities of all users and their assigned attributes are known, it can still be difficult to efficiently calculate the resulting set of permissions for a given user as all objects would need to be checked against all relevant policies.

To date, this has largely been addressed with hybrid ABAC models that use attributes simply for role assignment [Al-Kahtani and Sandhu 2002; Shafiq et al. 2005; Jin and Fang-chun 2006; Cirio et al. 2007; Cruz et al. 2009; Zhu et al. 2008; Wei et al. 2010] (allowing administrators to at least know what roles grant what permissions) or to put constraints on the permissions assigned to a role [Ferraiolo et al. 2001; Ge and Osborn 2004; Giuri and Iglio 1997; Abdallah and Khayat 2005; Lupu and Sloman 1997] (favoring an identity-based approach). As these methods use hybrid strategies, they come with the disadvantages of the hybrid models they use (i.e., namely, loss of flexibility and identity-less access control). ABAM [Zhang et al. 2005] is one of the few "pure" ABAC models that provides some level of auditability by restricting subjects to only possibly being assigned permissions in a predefined access matrix; however, it accomplishes this at the cost of being identity-less and requires users to be known and properly labelled in the access matrix.

It is important that more complete and efficient methods of auditing "pure" ABAC systems be developed to enable administrators to demonstrate compliance with specific regulations and directives that require before the fact auditing. Without this ability, ABAC will likely be unusable in cases where legal or industry regulations prohibit systems that rely solely on after the fact auditing techniques.

6.5. Separation of Duties

Separation of duties (SoD) is the notion that multiple persons should be required to complete a sensitive task to limit the potential for both error and fraud. In RBAC, this is supported through static SoD, where subjects are prohibited from being assigned conflicting roles, and dynamic SoD, where subjects are prohibited from activating conflicting roles in the same session [Ferraiolo et al. 2001]. However, in ABAC, application of this concept has been largely unexplored and left to future work. It is still unclear to what or how SoD type constraints might be applied to ABAC models and if additional constraints beyond those possible through policy languages are required.

Alipour and Sabbari [2012] attempt to solve this problem by introducing "can't_perform" rules that restrict a subject from performing certain actions (operations) on specified resources. This solution is problematic; however, in that it requires knowledge of both the subject and their possible conflicts of interest beforehand. Bijon

et al. [2013] propose an attribute-based constraint specification language (ABCL) that allows constraints to be placed on both attributes and attribute assignments. They demonstrate how this language may be used to specify SoD style constraints and validate its usefulness through a number of use cases. While this work may be part of a viable solution, it merely defines a language for representing constraints and lacks a formal model or framework for their use. Finally, a common solution is to use the SoD constraints from RBAC in hybrid ABAC models that include roles [Jin et al. 2012b; Shafiq et al. 2005; Cirio et al. 2007; Wei et al. 2010; Han et al. 2009]. However, as with other uses of hybrid ABAC, it comes at the cost of flexibility or the identity-less nature of ABAC.

6.6. Delegation

Delegation is a frequently desired access control feature that allows one subject to temporarily delegate their access rights to a more junior (in terms of access rights) subject. In RBAC research [Barka and Sandhu 2000a, 2000b], this is often accomplished by enabling a delegation of assigned roles under certain predefined constraints and revocation conditions, but it has also been expressed in terms of partial permission delegation [Wang and Osborn 2011; Zhang et al. 2003; Wang and Osborn 2006], in which a delegator creates and delegates a temporary role composed of a subset of their delegatable permissions. While delegation has been partially addressed in terms of attribute-based encryption [Waters 2011; Servos et al. 2013], few efforts to date have been made to apply a delegation model to ABAC.

Such a model of delegation could be applied to both delegation of attributes between users and delegation of resulting permissions granted by policies. Delegation of attributes could be partially supported through the use of X. 509 attribute certificates [Farrell and Housley 2002; Farrell et al. 2010]; however, this requires potentially lengthy certificate chains to be transmitted as part of a user's attribute-based credentials and could also lead to privacy concerns when sensitive attributes are involved. Moreover, attribute certificates are largely an implementation detail rather than a formal part of a delegation model. Dynamic delegation of permissions is more complex as attribute values (particularly for environment attributes, like time) may frequently change resulting in different permission assignments. Allowing delegation of granted permissions may require constant evaluation of relevant policies to ensure permissions are revoked when the delegator's access is removed due to a change in attributes, an approach that is both complicated and inefficient.

6.7. Attribute Storage and Sharing

When multiple attribute sources are used in an ABAC system (e.g., using attribute authorities from different organizations in a distributed system) complications can arises in terms of both evaluating the trustworthiness of attributes and ensuring that differing attribute sources are using compatible attributes (e.g., using the same namespace and data type for common attributes). The issue of trustworthiness is often dealt with by relying on pre-existing trust relations negotiated between organizations before access control takes place; however, in peer-to-peer scenarios this can be vastly more complicated. Shafiq et al. [2005] offer a potential solution in their hybrid ABAC model that includes a trust evaluation and negotiation framework that both provides a trust assessment of claimed attributes and a means to dynamically establish trust between collaborating organizations. Lee et al. [2008] propose an "attribute aggregation architecture" where attributes are gathered from neighboring peers and evaluated using a reputation-based trust scheme in which "each peer decides its reputation about other

peers based on its own experiences, and the trustworthiness of a peer is evaluated with the assist of aggregated reputation." It is possible that Shafiq and Lee's other research in dynamic trust negotiation could be easily applied to "pure" ABAC models; however, most work in this area has assumed attributes are derived from a trusted source.

Ensuring attributes from different sources are compatible would likely require a commonly accepted namespace or ontology of attribute names, or alternatively, some means of mapping attributes to equivalent representations (as suggested in Hu et al. [2013]). For example, if one organization's attribute store uses the name "job_title" and another "role" to describe the same attribute, it would be difficult to create policies that are applicable to members of both organizations without a detailed mapping between the two sets of attributes or complex policies that take into account the differences in attribute composition in each store. A secondary issue in attribute sharing is ensuring the confidentiality of sensitive user attributes. This is particularly a concern when ABAC systems are used in domains such as health care where leaking attributes about a user or object could be potentially compromising. Current work related to attribute privacy or confidentiality has largely been limited to attribute-based encryption applications, but some efforts have been made toward generic privacy preserving attribute sharing protocols [Camenisch et al. 2010; Ardagna et al. 2010; Esmaeeli and Shahriari 2010; Zhang et al. 2013].

6.8. Scalability

One of the important considerations before adopting ABAC as described in the NIST Guide to ABAC Definition and Considerations [Hu et al. 2013] is the scalability of ABAC systems. Unlike traditional access control technologies, such as RBAC, that have a proven track record in being adopted in large scale real-world systems, ABAC is still largely unproven in terms of practical scalability. ABAC requires complex interactions between access control components that may be distributed among different network resources or even across organizational boundaries. In large systems with thousands of users, permissions, and policies, it is unclear how manageable ABAC solutions would be, both in terms of administration and physical computing resources required. Real-world case studies of large scale systems utilizing ABAC concepts are required to determine the feasibility and usability of ABAC in such scenarios.

6.9. Administration and User Comprehension

A frequently overlooked aspect of ABAC is the "human aspect" or how usable such systems may be for users, access control administrators, and policy engineers. Lee and Winslett [2006] discuss the human factor challenges related to ABAC solutions and identify a number of open problems in ABAC research related to administration and usability. They describe the three main challenges as "Access Control Comprehension," "Technology Management," and "Policy Specification and Maintenance."

Lee and Winslett characterize "Access Control Comprehension" as the end user's ability to comprehend the access control decisions made regarding their access requests. In the classical models, access control decisions are relatively straightforward (e.g., in RBAC, users are either members of a role with the effective permissions they desire or not). However, in ABAC, decisions may be the result of complex policies that not only involve the attributes of the user but attributes of other, frequently changing, access control entities. Without sufficient understanding of both ABAC and the existing policies contained in the system, access decisions may seem arbitrary if not entirely magical from an end user perspective. Lee and Winslett point to efforts by Yao et al. [2005] toward visualization of such decisions as a first step toward a potential solution.

"Technology Management" concerns a user's ability to manage their access control credentials. In ABAC, subject credentials can be rather complex, consisting of technologies such as cryptographic credentials, X.509 certificates, and attribute sources from multiple distributed attribute stores. Lee and Winslett point to the research by Whitten and Tygar [1999] in which users had extreme difficulty managing PGP certificates for signing and encrypting e-mails to argue that end users of ABAC systems will have similar if not more extreme difficulties. This burden is worsened in systems that rely on end users to select the subset of attributes to be activated in a given session. While the solution to this problem likely lies in automating credential management, this has been largely unexplored in relation to ABAC and warrants further study.

"Policy Specification and Maintenance" addresses challenges related to the increased complexity inherent in ABAC administration and policy engineering. To date, almost no ABAC models provide complete (or even partial) administration models while at the same time requiring administrations and engineers to provide policies composed in complex XML-based policy languages such as XACML. Furthermore, the potentially distributed nature of ABAC means administration is no longer centralized but divided among multiple policy administration points and attribute stores. This significantly raises the training and education requirements for competent administrative users as well as hindering their ability to review current configurations for security issues. Potential solutions may be found in analysis tools that allow users with limited knowledge of mathematical or Boolean logic to create and evaluate realistic access control policies, in automated tools for mining ABAC policies [Xu and Stoller 2013, 2014, 2015] and in new administrative access control structures such as hierarchical attribute user and object groups [Servos and Osborn 2014].

6.10. Formal Security Analysis

While a number of works have sought to provide tools to analyze the security and safety of the traditional models (namely RBAC) [Li and Tripunitara 2006; Sasturkar et al. 2006; Stoller et al. 2007] and the policies they enforce, similar efforts for ABAC are still in their infancy. The most relevant efforts to date (e.g., Bryans [2005], Lin et al. [2010], Fisler et al. [2005], and Kolovski et al. [2007]) have focused on reasoning about and analyzing access control policies that may support attribute-based concepts independently of a formal access control model (e.g., policies written in XACML). Although many of these concepts and tools can be applied to the policies supported by ABAC models (particular if they are in a standardized policy language like XACML), they alone can not provide a full security analysis of a given ABAC model without taking into consideration the properties of the underlying model and the way in which policies are combined and enforced.

A sensible starting point for future ABAC focused security analysis work may be found in adapting the techniques used for RBAC such as those employed by Li and Tripunitara [2006]. Li and Tripunitara use security analysis techniques [Li and Winsborough 2003] to view RBAC as a state-transition system in which state changes occur via administrative operations, with the goal of determining if undesirable states are possible. Whilst they primarily use this state-transition system to explore security problems resulting from RBAC administration, a number of the queries they define on a given system state could be adapted for analyzing ABAC systems. In particular, the following queries could be of interest if attributes are considered in place of roles:

Simple Safety. If a state exists where a given (presumably untrusted) user can gain membership in a given role only intended for trusted users. A negative result would imply that the system is safe.

Simple Availability. If a given permission is attainable in every possible state to a given (presumably trusted) user. A positive result would imply that the permission is always available to the user.

Bounded Safety. If in every possible state, only a given subset of (presumably trusted) users can obtain the given permissions. A positive result would imply that the system is safe.

Liveness. Whether a given permission is always accessible to at least one user. A negative result (i.e., that the permission is always accessible) would imply the liveness of the permission holds in the system.

Mutual Exclusion. If there exists no possible state where a user can be a member of two distinct roles (r_1 and r_2). A positive result would imply that roles r_1 and r_2 are mutually exclusive.

Containment. Whether in every reachable state any user who has a given permission is a member of a given role. A positive result would imply that safety property is held (i.e., that all holders of a given permission are also in a given role) and an availability property is held (i.e., that a given permission is available to all members of a given role).

Adapting such queries to ABAC systems is challenging due to the increased flexibility provided by attribute-based policies and its identity-less nature in which users may not be known until runtime. This poses similar problems as faced when auditing ABAC systems (as discussed in Section 6.4), namely, that efficiently calculating the result of such queries is difficult when a large number of policies and attributes are present in a system. Rather than simply considering system states created by a combination of users, roles, and permissions (as in RBAC), analysis of ABAC system would have to account for all possible combinations of attributes (including possible combinations of values for each individual attribute), policies, and permissions, leading to a drastically larger state space.

7. CONCLUSIONS AND FUTURE WORK

This article has introduced a taxonomy of current areas of ABAC and PBAC research, provided a literature review of current attempts at formalizing ABAC models, and identified a number of open problems in the literature. The taxonomy introduced in Section 4 subdivides the current body of ABAC related research into related categories that are useful when discussing and comparing recent efforts. The review of "pure" and hybrid ABAC models in Section 5 provides one of the most comprehensive summaries of existing academic work toward ABAC model creation and has proven useful in identifying a number of areas for future work. The open problems examined in Section 6 serve as potential starting points for new research efforts.

As the literature surveyed in this work covered a number of different types of ABAC models in breadth, there is still room for future survey efforts directed at covering specific categories or aspects of models in depth. An in-depth comparison and analysis of how current models represent attribute-based policies, for example, would be beneficial to the community, as would a more in-depth look at a specific subcategory of models (e.g., a longer review of Pure General ABAC Models). Reviews of non-model related attribute topics could also be of interest, such as attribute mining, attribute storage and sharing, attribute confidentiality, and supporting model independent architectures.

APPENDIX

Table II. Column Legend for Tables III and IV

Column	Description
Object Attributes	Whether the model supports object or resource attributes.
User Attributes	Whether the model supports user or subject attributes.
Environment Attributes	Whether the model supports attributes that describe the systems environment (e.g., current time, number of online users).
Connection Attributes	If the model supports attributes relating to the subject's session and/or connection to the system (e.g., subject's host name, IP, session id).
Mutable Attributes	If the model supports attributes whose value change as a result of a subject's requests on a system.
Policy Language	If the model formalizes its own policy language (✓) or the language being used (e.g., XACML).
Hierarchical	Whether the model supports hierarchical constructs to simplify administration and/or increase flexibility of policies (e.g., hierarchical attributes, hierarchical user or object groups).
Recursive Rules	If the policy language presented in the work supports recursive rules or policies.
Trust	Whether the model incorporates the notion of trust similar to that found in trust-based access control.
User & Object Groups	Whether the model supports user or object groups to simplify administration and/or increase flexibility of policies.
Separation of Duties	Whether the model supports any kind of separation of duties and the types supported (e.g., static, dynamic).
Delegation	If subjects are able to delegate a subset of their attributes or privileges to other subjects.
Functional Specification	Whether a functional specification is provided with the model.
Formal Model	If the model is formalized (i.e., if any formal language or notation is used to fully describe the model).
Emulates Traditional Models	If it is shown that the model can emulate the traditional models of access control (e.g., DAC, MAC, RBAC).
Administration Model	If an administrative model or functions are defined or presented.
Complete Model	Whether the model is complete, that is, if all necessary components of a usable ABAC model are presented and described.
Extends	The models extended or used as the basis to create this hybrid ABAC model (only used in Table VI).
Identity-less	If this hybrid ABAC model allows for identity-less access control. That is, access control that does not require pre-existing knowledge about the user or their roles in the system (only used in Table VI).

Table III. Comparison of General ABAC Models

	[Wang et al. 2004]	[Jin et al. 2012a]	[Zhang et al. 2005]	[Rubio-Medrano et al. 2013]	[Servos and Osborn 2014]	[Ferraiolo et al. 2011]
Object Attributes	✗	✓	✓	✓	✓	✓(Attributes do not have values)
User Attributes	✓	✓	✓	✓	✓	✓(Attributes do not have values)
Environment Attributes	✗	✗	✗	✓	✓	✗
Connection Attributes	✗	✗	✗	✗	✓	✗
Mutable Attributes	✗	✗	✗	✗	✗	✗
Policy Language	Has method of representing policies but no defined language	✓	No details given for how policies are represented	No policy language use (left to future work)	✓	Policies expressed as chain of attribute assignments
Hierarchical	Hierarchical attributes	✗	✗	✗	Hierarchical user and object groups	Hierarchical attributes
Recursive Rules	✓	✗	✗	Supported via cycles in the TP-Graph	✗	✗
Trust	✗	✗	✗	✗	✗	✗
User & Object Groups	✗	✗	✗	✗	✓	✗
Separation of Duties	✗	✗	✗	✗	✗	✓
Delegation	✗	✗	✗	✗	✗	✗
Functional Specification	✗	✓	✗	✗	✗	✗
Formal Model	✓	✓	✓	Largely informal	✓	✓
Emulates Traditional Models	Not demonstrated	✓	Not demonstrated	Not demonstrated	✓	✓
Administration Model	✗	Limited	Very limited	✗	✗	✓
Complete Model	Only models policies and their evaluation	✓	✓	✓	✓	✓

Table IV. Comparison of Domain Specific ABAC Models

	[Buehrer and Wang 2012]	[Burmester et al. 2013]	[Smari et al. 2009, 2014]	[Liang et al. 2012]	[Covington and Sastry Covington and Sastry]	[Kerschbaum 2010]	[Lang et al. 2006, 2009]
Domain	Cloud Computing	Real-time Systems	Collaborative Environments	Collaborative Environments	Mobile Environments	Mobile Environments	Grid computing
Object Attributes	✓	✓	✓	✓	✓	✓	✓
User Attributes	✓	✓	✓	✓	✓	✓	✓
Environment Attributes	✓	✓	✗	✓	✓	✗	✓
Connection Attributes	✗	✗	✗	✗	✗	✗	Shown in example but not model
Mutable Attributes	✗	✗	Mutable trust attribute	✗	Limited, based on transaction attributes	✗	✗
Policy Language	Class Algebra (from Cadabia knowledge base)	Does not mention policies	Policy language shown in examples but not defined	XACML	Claims to have policy language but is left undefined and no examples given	XACML	Policies are algorithms, no language used/defined
Hierarchical	✗	✗	✗	✗	✗	✗	✗
Recursive Rules	✗	✗	✗	✗	✗	✗	✗
Trust	✗	✗	✓	✗	✗	✗	✗
User & Object Groups	✗	✗	✗	✗	✗	✗	✗
Separation of Duties	✗	✗	✗	✗	✗	✗	✗
Delegation	✗	✗	✗	✗	✗	✗	✗
Functional Specification	✗	✗	✗	✗	✗	✗	✗
Formal Model	Informal	Only formalizes real-time attributes and packet mechanics	✓	✓	Informal	✓	✓
Emulates Traditional Models	Not demonstrated	Not demonstrated	Not demonstrated	Not demonstrated	Not demonstrated	Not demonstrated	Not demonstrated
Administration Model	✗	✗	✗	✗	✗	✗	✗
Complete Model	Lacks details, mostly describes policy use	Only models real-time attributes and packet mechanics	Lacks details, unclear how policies are evaluated and format of attributes	Lacks details, more architecture then model	Lacks details, policy language not formalized	✓	✓

Table V. Comparison of Domain Specific ABAC Models (continued from Table IV)

	[Lang et al. 2010]	[Yuan and Tong 2005]	[Shen and Hong 2006]	[Dan et al. 2012]	[Xia and Liu 2009]	[Shen 2009]	[Zhang et al. 2014]
Domain	Grid computing	Web Services	Web Services	Web Services	Web Services	Web Services	Web Services
Object Attributes	✓	✓	✓	✓	✓	✗	✓
User Attributes	✓	✓	✓	✓	✓	✓	✓
Environment Attributes	✓	✓	✓	✓	✓	✗	✓
Connection Attributes	✗	✗	✗	✗	✗	✗	✗
Mutable Attributes	✗	✗	✗	✗	✗	✗	✗
Policy Language	XACML	Model lacks language, implementation uses XACML	XACML	Model lacks language, implementation uses XACML	XACML	XACML	XACML
Hierarchical	✗	✗	✗	✗	✗	✗	✗
Recursive Rules	✗	✗	✗	✗	✗	✗	✗
Trust	✗	✗	✗	✗	✗	✗	Claims trust attribute but fails to provide details
User & Object Groups	✗	✗	✗	✗	✗	✗	✗
Separation of Duties	✗	✗	✗	✗	✗	✗	✗
Delegation	✗	✗	✗	✗	✗	✗	✗
Functional Specification	✗	✗	✗	✗	✗	✗	✗
Formal Model	Largely informal	Simplistic	Simplistic	Simplistic	✓	Informal	Largely Informal
Emulates Traditional Models	Not demonstrated	Not demonstrated	Not demonstrated	Not demonstrated	Not demonstrated	Not demonstrated	Not demonstrated
Administration Model	✗	✗	✗	✗	✗	✗	✗
Complete Model	Minimal model, mostly architecture combining existing works	✓	✓	More implementation using XACML then model	✓	More theoretical architecture combining existing works then model	Basic definitions for model, mostly architecture combining existing works.

Table VI. Comparison of Hybrid ABAC Models

	Parameterized Role-Based Access Control					Attribute-Based Role Assignment	
	[Ge and Osborn 2004]	[Giuri and Iglio 1997]	[Abdallah and Khayat 2005]	[Lupu and Sloman 1997]	[Fischer et al. 2009]	[Al-Kahtani and Sandhu 2002]	[Shafiq et al. 2005]
Extends	Role Graph Model [Nyanchama and Osborn 1999]	RBAC	FRBAC [Khayat and Abdallah 2003]	RBAC & RBM	RBAC	RBAC	GTRBAC [Joshi et al. 2005]
Identity-less	✗	✗	✗	✗	✗	✓	Both
Object Attributes	✓	✗	✓	✓	✗	✗	✗
User Attributes	✓	✓	✓	✗	✓	✓	✓
Environment Attributes	✗	Day of week attribute shown in example but not detailed	✗	Time attribute shown in example but not detailed	✗	✗	Temporal attributes from extended model
Connection Attributes	✗	✗	✗	✗	✗	✗	✗
Mutable Attributes	✗	✗	✗	✗	✗	✗	Mutable trust values
Policy Language	XPath	No policy language formally defined (shown in examples)	N/A	No policy language formally defined (shown in examples)	✓	✓	SAML & X-GTRBAC [Bhatti et al. 2005]
Hierarchical	Hierarchical roles	✗	✗	Hierarchical roles	✗	Hierarchical roles	Hierarchical roles
Trust	✗	✗	✗	✗	✗	✗	✓
Separation of Duties	From extended model	✗	✗	✗	✗	Constraints on use of roles mentioned but not detailed	From extended model
Delegation	✗	✗	✗	✗	✗	✗	✗
Functional Specification	✗	✗	✗	✗	✗	✗	✗
Formal Model	✓	✓	✓	Informal	✓	✓	✓
Administration Model	Does not expand on extended model	✗	✗	✗	✗	✗	✗
Complete Model	✓	Definition and evaluation of policies and attributes is only vaguely defined	✓	Lacks details, mostly framework for adding RBM concepts to RBAC	✓	✓	✓

Table VII. Comparison of Hybrid ABAC Models (continued from Table VI)

	Attribute-Based Role Assignment (continued)					
	[Jin and Fang-chun 2006]	[Cirio et al. 2007]	[Cruz et al. 2009, 2008]	[Zhu et al. 2008]	[Wei et al. 2010]	[He et al. 2011]
Extends	RBAC	RBAC	RBAC	RBAC	RBAC	RBAC
Identity-less	✓	✓	Both	✓	✓	Both
Object Attributes	✗	✗	✓	✓	✓	✗
User Attributes	✓	✓	✓	✓	✓	✓
Environment Attributes	Temporal attributes	✗	✗	✗	✗	✗
Connection Attributes	✗	✗	✗	✗	✗	✗
Mutable Attributes	✗	✗	✗	✗	✗	✗
Policy Language	ALC(D) [Baader and Hanschke 1991]	Unclear. OWL and SPARQL [PrudHommeaux and Seaborne 2008] used for modeling RBAC.	OWL [McGuinness et al. 2004]	Policy language not formally defined	No policy language shown or defined	SWRL [Horrocks et al. 2004]
Hierarchical	Hierarchical roles	✗	Hierarchical roles	Hierarchical roles	Hierarchical roles	Hierarchical roles
Trust	✗	✗	✗	✗	✗	✗
Separation of Duties	✗	✓	✗	✗	✓	✓
Delegation	✗	✗	✗	✗	✗	✗
Functional Specification	✗	✗	✗	✗	✗	✗
Formal Model	✓	Only RBAC modeling formalized	Largely informal	✓	✓	✓
Administration Model	✗	✗	✗	✗	✗	✗
Complete Model	✓	Mostly covers modeling RBAC in a an OWL-DL ontology. Few details given on attributes.	✓	✓	Limited details on how constraints and policies are handled or defined	✓

Table VIII. Comparison of Hybrid ABAC Models (continued from Table VII)

	Attribute-Centric	**Role-Centric**	**Unified Models of Access Control**		
	[Huang et al. 2012]	[Jin et al. 2012b]	[Han et al. 2009]	[Che et al. 2010]	[Cheng et al. 2014]
Extends	RBAC & ABAC	NIST RBAC [Ferraiolo et al. 2001] & $ABAC_\alpha$ [Jin et al. 2012a]	RBAC, TBAC, & ABAC	ABAC & BBAC	ABAC & UURAC [Cheng et al. 2012]
Identity-less	✓	✗	Both	✓	✗
Object Attributes	✓	✓	✓	✗	✓
User Attributes	✓	✓	✓	✓	✓
Environment Attributes	✓	✗	✗	✓	✗
Connection Attributes	✗	✗	✗	✗	✗
Mutable Attributes	✗	✗	✗	Limited. Based on user behaviors.	✗
Policy Language	Informal custom policy language	CPL [Jin et al. 2012a]	XACML	Example policies shown but no language defined.	✓
Hierarchical	✗	Hierarchical roles from NIST RBAC	Hierarchical roles	✗	✗
Trust	✗	✗	✗	✗	✗
Separation of Duties	✗	From NIST RBAC	✓	✗	✗
Delegation	✗	✗	✗	✗	✗
Functional Specification	✗	✓	✗	✗	✗
Formal Model	✓ (other than policy)	✓	✓	Largely informal	✓
Administration Model	✗	From NIST RBAC	✗	✗	✗
Complete Model	✓	✓	✓	✓	✓

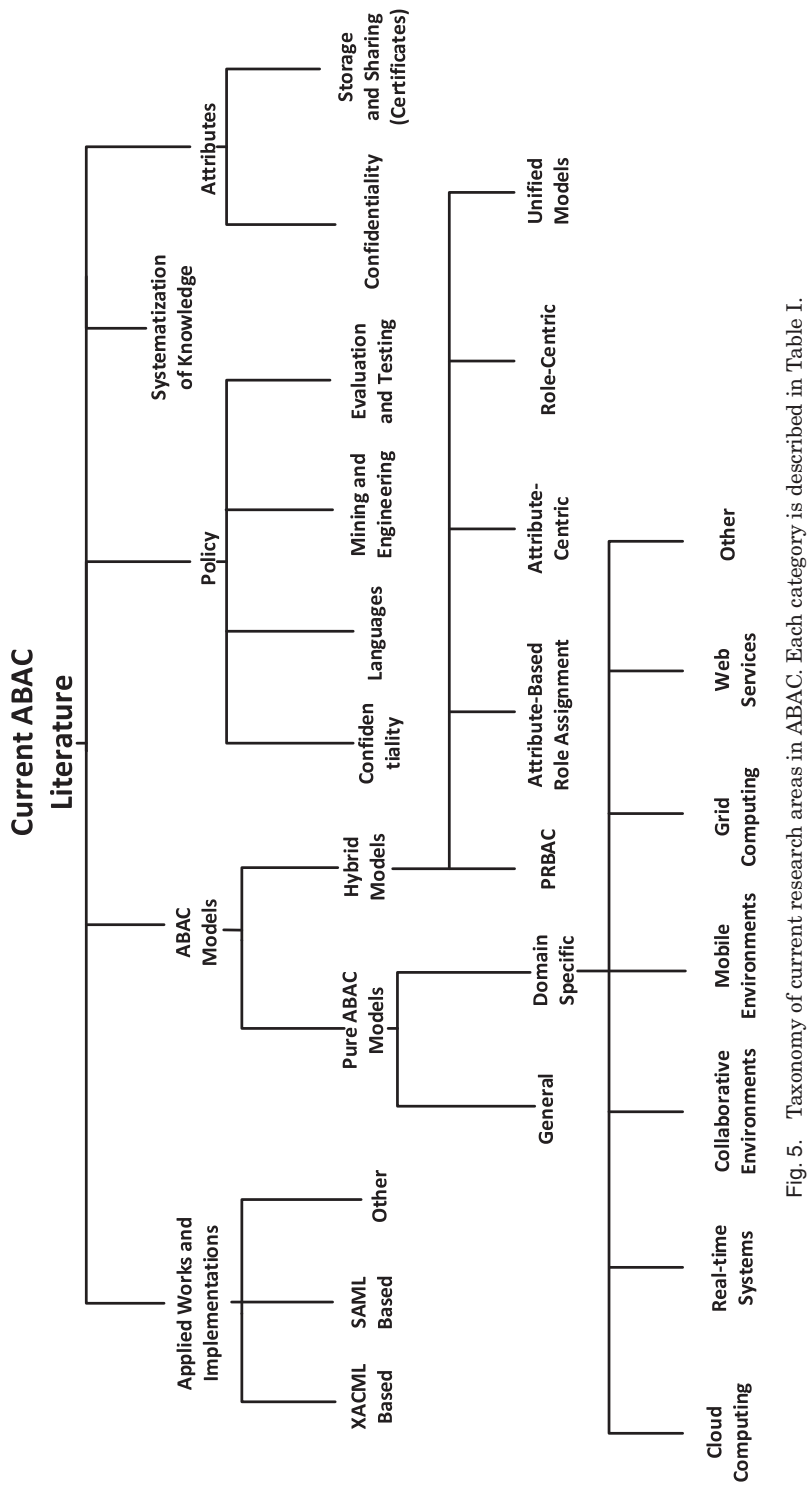


Fig. 5. Taxonomy of current research areas in ABAC. Each category is described in Table I.

REFERENCES

Ali E. Abdallah and Etienne J. Khayat. 2005. A formal model for parameterized role-based access control. In *Formal Aspects in Security and Trust*. Springer, 233–246.

Nabil R. Adam, Vijayalakshmi Atluri, Elisa Bertino, and Elena Ferrari. 2002. A content-based authorization model for digital libraries. *IEEE Transactions on Knowledge and Data Engineering* 14, 2 (2002), 296–315.

Mohammad A. Al-Kahtani and Ravi Sandhu. 2002. A model for attribute-based user-role assignment. In *Proceedings of the 2002 18th Annual Computer Security Applications Conference.* IEEE, 353–362.

Hadiseh Seyyed Alipour and Mehdi Sabbari. 2012. Definition of action and attribute based access control rules for web services. In *Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management*. 869–878.

Claudio Agostino Ardagna, Sabrina De Capitani di Vimercati, Gregory Neven, Stefano Paraboschi, F.-S. Preiss, Pierangela Samarati, and Mario Verdicchio. 2010. Enabling privacy-preserving credential-based access control with XACML and SAML. In *Proceedings of the 2010 IEEE 10th International Conference on Computer and Information Technology (CIT'10)*. IEEE, 1090–1095.

Franz Baader and Philipp Hanschke. 1991. *A Scheme for Integrating Concrete Domains into Concept Languages*. Technical Report RR-91-10. DFKI Deutsches Forschungszentrum fr Knstliche Intelligenz.

Ezedin Barka and Ravi Sandhu. 2000a. Framework for role-based delegation models. In *Proceedings of the 16th Annual Conference on Computer Security Applications (ACSAC'00)*. IEEE, 168–176.

Ezedin Barka and Ravi Sandhu. 2000b. A role-based delegation model and some extensions. In *Proceedings of the 23rd National Information Systems Security Conference*. 396–404.

Steve Barker. 2009. The next 700 access control models or a unifying meta-model? In *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies*. ACM, 187–196.

John Bethencourt, Amit Sahai, and Brent Waters. 2007. Ciphertext-policy attribute-based encryption. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP'07)*. IEEE, 321–334.

Rafae Bhatti, Arif Ghafoor, Elisa Bertino, and James BD Joshi. 2005. X-GTRBAC: An XML-based policy specification framework and architecture for enterprise-wide access control. *ACM Transactions on Information and System Security (TISSEC)* 8, 2 (2005), 187–227.

Khalid Zaman Bijon, Ram Krishman, and Ravi Sandhu. 2013. Constraints specification in attribute based access control. *Science* 2, 3 (2013), pp–131.

Rakesh Bobba, Omid Fatemieh, Fariba Khan, Arindam Khan, Carl A. Gunter, Himanshu Khurana, and Manoj Prabhakaran. 2010. Attribute-based messaging: Access control and confidentiality. *ACM Transactions on Information and System Security (TISSEC)* 13, 4 (2010), 31.

David F. C. Brewer and Michael J. Nash. 1989. The Chinese wall security policy. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*. IEEE, 206–214.

Jery Bryans. 2005. Reasoning about XACML policies using CSP. In *Proceedings of the 2005 Workshop on Secure Web Services*. ACM, 28–35.

Daniel J. Buehrer, Lo Tse-Wen, and Hsieh Chih-Ming. 2001. Abia cadabia: A distributed, intelligent database architecture. *Intelligent Multimedia, Computing, and Communications* (2001), 1–3.

Daniel J. Buehrer and Chun-Yao Wang. 2012. CA-ABAC: Class algebra attribute-based access control. In *Proceedings of the 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 03*. IEEE Computer Society, 220–225.

Mike Burmester, Emmanouil Magkos, and Vassilis Chrissikopoulos. 2013. T-ABAC: An attribute-based access control model for real-time availability in highly dynamic systems. In *Proceedings of the 2013 IEEE Symposium on Computers and Communications (ISCC'13)*. IEEE, 000143–000148.

Jan Camenisch, Sebastian Mödersheim, Gregory Neven, Franz-Stefan Preiss, and Dieter Sommer. 2010. A card requirements language enabling privacy-preserving access control. In *Proceedings of the 15th ACM Symposium on Access Control Models and Technologies*. ACM, 119–128.

David W. Chadwick, Alexander Otenko, and Edward Ball. 2003. Role-based access control with X.509 attribute certificates. *Internet Computing, IEEE* 7, 2 (2003), 62–69.

Yanzhe Che, Qiang Yang, Chunming Wu, and Lianhang Ma. 2010. BABAC: An access control framework for network virtualization using user behaviors and attributes. In *Proceedings of the 2010 IEEE/ACM International Conference on Green Computing and Communications & International Conference on Cyber, Physical and Social Computing*. IEEE Computer Society, 747–754.

Yuan Cheng, Jaehong Park, and Ravi Sandhu. 2012. A user-to-user relationship-based access control model for online social networks. In *Data and Applications Security and Privacy XXVI*. Springer, 8–24.

Yuan Cheng, Jaehong Park, and Ravi Sandhu. 2014. Attribute-aware relationship-based access control for online social networks. In *Data and Applications Security and Privacy XXVIII*. Springer, 292–306.

Lorenzo Cirio, Isabel F Cruz, and Roberto Tamassia. 2007. A role and attribute based access control system using semantic web technologies. In *Proceedings of the 2007 OTM Confederated International Conference on On the Move to Meaningful Internet Systems - Volume Part II (OTM'07)*. Springer, 1256–1266.

James Clark and Steve DeRose. 1999. XML path language (XPath). *W3C Recommendation* 16.

Michael J. Covington and Manoj R. Sastry. A contextual attribute-based access control model. In *Proceedings of the 2006 International Conference on On the Move to Meaningful Internet Systems: AWeSOMe, CAMS, COMINF, IS, PKSinBIT*.

Isabel F. Cruz, Rigel Gjomemo, Benjamin Lin, and Mirko Orsini. 2008. A location aware role and attribute based access control system. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 84.

Isabel F. Cruz, Rigel Gjomemo, Benjamin Lin, and Mirko Orsini. 2009. A constraint and attribute based security framework for dynamic role assignment in collaborative environments. In *Collaborative Computing: Networking, Applications and Worksharing*. Springer, 322–339.

Ni Dan, Shi Hua-Ji, Chen Yuan, and Guo Jia-Hu. 2012. Attribute based access control (ABAC)-based cross-domain access control in service-oriented architecture (SOA). In *Proceedings of the 2012 International Conference on Computer Science & Service System (CSSS'12)*. IEEE, 1405–1408.

Agostino Dovier, Carla Piazza, Enrico Pontelli, and Gianfranco Rossi. 2000. Sets and constraint logic programming. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 22, 5 (2000), 861–931.

Ali Esmaeeli and Hamid Reza Shahriari. 2010. Privacy protection of grid service requesters through distributed attribute based access control model. In *Proceedings of the 5th International Conference on Advances in Grid and Pervasive Computing*. Springer, 573–582.

S. Farrell and R. Housley. 2002. *An Internet Attribute Certificate Profile for Authorization*. RFC 3281. RFC Editor. Retrieved from https://www.ietf.org/rfc/rfc3281.txt.

S. Farrell, R. Housley, and S. Turner. 2010. *An Internet Attribute Certificate Profile for Authorization*. RFC 5755. RFC Editor. Retrieved from https://tools.ietf.org/html/rfc5755.

David Ferraiolo. 2013. Towards an ABAC Family of Models. Retrieved from http://csrc.nist.gov/projects/abac/july2013_workshop/july2013_abac_workshop_abac-model-framework_dferraiolo.pdf.

David Ferraiolo, Vijayalakshmi Atluri, and Serban Gavrila. 2011. The policy machine: A novel architecture and framework for access control policy specification and enforcement. *Journal of Systems Architecture* 57, 4 (2011), 412–424.

David Ferraiolo, Serban Gavrila, and Wayne Jansen. 2015. *Policy Machine: Features, Architecture, and Specification*. Technical Report NISTIR 7987 Revision 1. National Institute of Standards and Technology. http://dx.doi.org/10.6028/NIST.IR.7987r1

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

Elena Ferrari, Nabil R. Adam, Vijayalakshmi Atluri, Elisa Bertino, and Ugo Capuozzo. 2002. An authorization system for digital libraries. *The VLDB Journal* 11, 1 (2002), 58–67.

Kathi Fisler, Shriram Krishnamurthi, Leo A. Meyerovich, and Michael Carl Tschantz. 2005. Verification and change-impact analysis of access-control policies. In *Proceedings of the 27th International Conference on Software Engineering*. ACM, 196–205.

Mei Ge and Sylvia L. Osborn. 2004. A design for parameterized roles. In *Research Directions in Data and Applications Security XVIII*. Springer, 251–264.

Luigi Giuri and Pietro Iglio. 1997. Role templates for content-based access control. In *Proceedings of the Second ACM Workshop on Role-Based Access Control*. ACM, 153–159.

Simon Godik, Anne Anderson, Bill Parducci, Polar Humenn, and Sekhar Vajjhala. 2002. *OASIS eXtensible Access Control Markup Language (XACML)*. Technical Report. OASIS.

Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. 2006. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*. ACM, 89–98.

Ruo-Fei Han, Hou-Xiang Wang, Qian Xiao, Xiao-Pei Jing, and Hui Li. 2009. A united access control model for systems in collaborative commerce. *Journal of Networks* 4, 4 (2009), 279–289.

Zhengqiu He, Lifa Wu, Huabo Li, Haiguang Lai, and Zheng Hong. 2011. Semantics-based access control approach for web service. *Journal of Computers* 6, 6 (2011), 1152–1161.

Richard Dean Holowczak. 1997. *Extractors for Digital Library Objects*. Ph.D. Dissertation. Rutgers University, Department of MS/CIS.

Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, and others. 2004. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member Submission* 21 (2004), 79.

Vincent C. Hu, David Ferraiolo, Rick Kuhn, Arthur R. Friedman, Alan J. Lang, Margaret M. Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen Scarfone. 2013. Guide to attribute based access control (ABAC) Definition and Considerations (Draft). *NIST Special Publication* 800 (2013), 162.

Jingwei Huang, David M. Nicol, Rakesh Bobba, and Jun Ho Huh. 2012. A framework integrating attribute-based policies into role-based access control. In *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies*. ACM, 187–196.

John Hughes and Eve Maler. 2005. *Security Assertion Markup Language (SAML) V2.0 Technical Overview*. Technical Report. OASIS.

Junbeom Hur and Dong Kun Noh. 2011. Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Transactions on Parallel and Distributed Systems* 22, 7 (2011), 1214–1221.

INCITS. 2013. *Information Technology - Next Generation Access Control - Functional Architecture (NGAC-FA)*. Technical Report INCITS 499-2013. American National Standard for Information Technology, American National Standards Institute.

INCITS. 2015. *Information technology - Next Generation Access Control Generic Operations and Data Structures (NGAC-GOADS)*. Technical Report INCITS 499-2013. American National Standard for Information Technology, American National Standards Institute.

Peng Jin and Yang Fang-chun. 2006. Description logic modeling of temporal attribute-based access control. In *Proceedings of the 2006 1st International Conference on Communications and Electronics*. IEEE, 414–418.

Xin Jin, Ram Krishnan, and Ravi Sandhu. 2012a. A unified attribute-based access control model covering DAC, MAC and RBAC. In *Data and Applications Security and Privacy XXVI*. Springer, 41–55.

Xin Jin, Ravi Sandhu, and Ram Krishnan. 2012b. RABAC: Role-centric attribute-based access control. In *Proceedings of the 6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security: Computer Network Security*. Springer, 84–96.

James B. D. Joshi, Elisa Bertino, Usman Latif, and Arif Ghafoor. 2005. A generalized temporal role-based access control model. *IEEE Transactions on Knowledge and Data Engineering* 17, 1 (2005), 4–23.

Florian Kerschbaum. 2010. An access control model for mobile physical objects. In *Proceedings of the 15th ACM Symposium on Access Control Models and Technologies*. ACM, 193–202.

Etienne J. Khayat and Ali E. Abdallah. 2003. A formal model for flat role-based access control. In *ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'03)*, Vol. 4.

Vladimir Kolovski, James Hendler, and Bijan Parsia. 2007. Analyzing web access control policies. In *Proceedings of the 16th International Conference on World Wide Web*. ACM, 677–686.

D. Richard Kuhn, Edward J. Coyne, and Timothy R. Weil. 2010. Adding attributes to role-based access control. *IEEE Computer* 43, 6 (2010), 79–81.

Bo Lang, Ian Foster, Frank Siebenlist, Rachana Ananthakrishnan, and Tim Freeman. 2006. Attribute based access control for grid computing. Retrieved from http://www.mcs.anl.gov/uploads/cels/papers/P1367.pdf.

Bo Lang, Ian Foster, Frank Siebenlist, Rachana Ananthakrishnan, and Tim Freeman. 2009. A flexible attribute based access control method for grid computing. *Journal of Grid Computing* 7, 2 (2009), 169–180.

Bo Lang, Hangyu Li, and Wenting Ni. 2010. Attribute-based access control for layered grid resources. In *Communication and Networking*. Springer, Berlin, 31–40.

Adam J. Lee and Marianne Winslett. 2006. Open problems for usable and secure open systems. In *Proceediings of the Workshop on Usability Research Challenges for Cyberinfrastructure and Tools Held in Conjunction with ACM CHI*.

Jaewon Lee, Heeyoul Kim, and Joon Sung Hong. 2008. An attribute aggregation architecture with trust-based evaluation for access control. In *Proceedings of the NOMS 2008-2008 IEEE Network Operations and Management Symposium*. 1011–1014.

Ninghui Li and Mahesh V. Tripunitara. 2006. Security analysis in role-based access control. *ACM Transactions on Information and System Security (TISSEC)* 9, 4 (2006), 391–420.

Ninghui Li and William H. Winsborough. 2003. Beyond proof-of-compliance: Safety and availability analysis in trust management. In *2003 Symposium on Security and Privacy*. IEEE, 123–139.

Feng Liang, Haoming Guo, Shengwei Yi, and Shilong Ma. 2012. A multiple-policy supported attribute-based access control architecture within large-scale device collaboration systems. *Journal of Networks* 7, 3 (2012), 524–531.

Dan Lin, Prathima Rao, Elisa Bertino, Ninghui Li, and Jorge Lobo. 2010. EXAM: A comprehensive environment for the analysis of access control policies. *International Journal of Information Security* 9, 4 (2010), 253–273.

Emil Lupu and Morris Sloman. 1997. Reconciling role based management and role based access control. In *Proceedings of the Second ACM Workshop on Role-Based Access Control*. ACM, 135–141.

Deborah L. McGuinness, Frank Van Harmelen, and Others. 2004. OWL web ontology language overview. *W3C Recommendation* (2004).

Matunda Nyanchama and Sylvia Osborn. 1999. The role graph model and conflict of interest. *ACM Transactions on Information and System Security (TISSEC)* 2, 1 (1999), 3–33.

Jaehong Park and Ravi Sandhu. 2004. The UCON ABC usage control model. *ACM Transactions on Information and System Security (TISSEC)* 7, 1 (2004), 128–174.

Eric PrudHommeaux and Andy Seaborne. 2008. SPARQL query language for RDF. *W3C Recommendation* 15 (2008).

Carlos E. Rubio-Medrano, Clinton D'Souza, and Gail-Joon Ahn. 2013. Supporting secure collaborations with attribute-based access control. In *Proceedings of the 2013 9th International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom)*. IEEE, 525–530.

Amit Sasturkar, Ping Yang, Scott D. Stoller, and C. R. Ramakrishnan. 2006. Policy analysis for administrative role based access control. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW'06)*. IEEE.

Daniel Servos. 2012. *A Role and Attribute Based Encryption Approach to Privacy and Security in Cloud Based Health Services*. Master's thesis. Lakehead University. Retrieved from http://knowledgecommons.lakeheadu.ca/handle/2453/286.

Daniel Servos, Sabah Mohammed, Jinan Fiaidhi, and Tai hoon Kim. 2013. Extensions to ciphertext-policy attribute-based encryption to support distributed environments. *International Journal of Computer Applications in Technology* 47, 2 (2013), 215–226.

Daniel Servos and Sylvia L. Osborn. 2014. HGABAC: Towards a formal model of hierarchical attribute-based access control. In *Proceedings of the 7th International Symposium on Foundations and Practice of Security (FPS'14)*. Springer, 187–204.

Basit Shafiq, Elisa Bertino, and Arif Ghafoor. 2005. Access control management in a distributed environment supporting dynamic collaboration. In *Proceedings of the 2005 Workshop on Digital Identity Management*. ACM, 104–112.

Haibo Shen. 2009. A semantic-aware attribute-based access control model for web services. In *Proceedings of the 9th International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 693–703.

Hai-bo Shen and Fan Hong. 2006. An attribute-based access control model for web services. In *Proceedings of the 2006 7th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06)*. IEEE, 74–79.

Waleed W. Smari, Patrice Clemente, and Jean-Francois Lalande. 2014. An extended attribute based access control model with trust and privacy: Application to a collaborative crisis management system. *Future Generation Computer Systems* 31 (2014), 147–168.

Waleed W. Smari, Jian Zhu, and Patrice Clemente. 2009. Trust and privacy in attribute based access control for collaboration environments. In *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services*. ACM, 49–55.

Scott D. Stoller, Ping Yang, C. R. Ramakrishnan, and Mikhail I. Gofman. 2007. Efficient policy analysis for administrative role based access control. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*. ACM, 445–455.

Guojun Wang, Qin Liu, and Jie Wu. 2010. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*. ACM, 735–737.

He Wang and Sylvia L. Osborn. 2006. Delegation in the role graph model. In *Proceedings of the 11th ACM Symposium on Access Control Models and Technologies*. ACM, 91–100.

He Wang and Sylvia L. Osborn. 2011. Static and dynamic delegation in the role graph model. *IEEE Transactions on Knowledge and Data Engineering* 23, 10 (2011), 1569–1582.

Lingyu Wang, Duminda Wijesekera, and Sushil Jajodia. 2004. A logic-based framework for attribute based access control. In *Proceedings of the 2004 ACM Workshop on Formal Methods in Security Engineering*. ACM, 45–55.

Brent Waters. 2011. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Proceedings of the International Workshop on Public Key Cryptography*. Springer, 53–70.

Yonghe Wei, Chunjing Shi, and Weiping Shao. 2010. An attribute and role based access control model for service-oriented environment. In *Proceedings of the 2010 Chinese Control and Decision Conference*. IEEE, 4451–4455.

Alma Whitten and J. Doug Tygar. 1999. Why Johnny can't encrypt: A Usability Evaluation of PGP 5.0. In *Usenix Security*, Vol. 1999.

Jian Shu Lianghong Shi Bing Xia and Linlan Liu. 2009. Study on action and attribute-based access control model for web services. In *Proceedings of the 2009 2nd International Symposium on Information Science and Engineering*. 213–216.

Zhongyuan Xu and Scott D. Stoller. 2013. Mining attribute-based access control policies from RBAC policies. In *Proceedings of the 10th International Conference and Expo on Emerging Technologies for a Smarter World (CEWIT'10)*. IEEE, 1–6.

Zhongyuan Xu and Scott D. Stoller. 2014. Mining attribute-based access control policies from logs. In *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 276–291.

Zhongyuan Xu and Scott D. Stoller. 2015. Mining attribute-based access control policies. *IEEE Transactions on Dependable and Secure Computing* 12, 5 (2015), 533–545.

Danfeng Yao, Michael Shin, Roberto Tamassia, and William H. Winsborough. 2005. Visualization of automated trust negotiation. In *Proceedings of the IEEE Workshop on Visualization for Computer Security (VizSEC'05)*. IEEE, 65–74.

Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. 2010. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *Proceedings of the 2010 IEEE of INFOCOM*. IEEE, 1–9.

Eric Yuan and Jin Tong. 2005. Attributed based access control (ABAC) for web services. In *Proceedings of the IEEE International Conference on Web Services (ICWS'05)*. IEEE, 569.

Guoping Zhang, Jing Liu, and Jianbo Liu. 2013. Protecting sensitive attributes in attribute based access control. In *Proceedings of the International Conference on Service-Oriented Computing (ICSOC'13)*. Springer, 294–305.

Xinwen Zhang, Yingjiu Li, and Divya Nalla. 2005. An attribute-based access matrix model. In *Proceedings of the 2005 ACM Symposium on Applied Computing*. ACM, 359–363.

Xinwen Zhang, Sejong Oh, and Ravi Sandhu. 2003. PBDM: A flexible delegation model in RBAC. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies*. ACM, 149–157.

Yongsheng S. Zhang, Mingfeng F. Wu, Lei Wu, and Yuanyuan Y. Li. 2014. Attribute-based access control security model in service-oriented computing. In *Proceedings of the 2012 International Conference on Cybernetics and Informatics*. Springer, 1473–1479.

Jian Zhu and Waleed W. Smari. 2008. Attribute based access control and security for collaboration environments. In *Proceedings of the 2008 IEEE National Aerospace and Electronics Conference*. IEEE, 31–35.

Yiqun Zhu, Jianhua Li, and Quanhai Zhang. 2008. General attribute based RBAC model for web services. *Wuhan University Journal of Natural Sciences* 13, 1 (2008), 81–86.

Received June 2015; revised July 2016; accepted October 2016