# Enabling Flexible Administration in ABAC Through Policy Review: *A Policy Machine Case Study*

Sherifdeen Lawal, Ram Krishnan
Univ. of Texas at San Antonio, Texas, United State
sherifdeen.lawal@utsa.edu, ram.krishnan@utsa.edu

***Abstract*—The Next Generation Access Control (NGAC), founded on the Policy Machine (PM), is a robust Attribute-Based Access Control (ABAC) framework that enables a structured and flexible approach for the establishment of the conventional access control models. The authorization state of the policy machine is represented as an annotated Directed Acyclic Graph (DAG). Structurally, relations among attributes of the same type are hierarchical, and creating new relation(s) to specify an authorization may be achieved through different approaches. However, one or more limitations can make most of the obvious approaches to grant new access inconsistent with existing rules. We proposed an algorithm that provides the PM administrator a comprehensive list of all possible approaches to authorize access. The approaches generated by our algorithm can help the PM administrator makes an informed decision before authorizing access.**

***Index Terms*—Attribute Based Access Control, Policy Review, Authorization, Policy Machine, Authorization Graph**

## I. Introduction

Attribute-Based Access Control (ABAC) remains to be a promising form of access control. Unlike traditional access control. ABAC authorization to perform a set of operations is determined by evaluating attributes associated with the subject, object, and in some cases, environmental conditions against policy, rules, or relationships that describe the allowable operations for a given set of attributes [13].

The NIST Next Generation Access Control (NGAC) is one of the two current implementations of ABAC. The Policy Machine (PM) is the basis for the NGAC and can manage policies easily. There is a considerable amount of research work in the literature, applying the Policy Machine (PM). Bhatt et al. [4], [11], utilizes the Policy Machine for an attribute-based extension of OpenStack access control and implements (rHGABAC), a restricted version of the Hierarchical Group Attribute-Based Access Control [16]. For achieving adaptive policies, efficient and easy policy management in IoT [8] and Mobile Health applications [12], NGAC specification has been utilized to authorize communication between devices.

Another application of NGAC specification is the centrally managed attribute-based access control policies for resource repositories distributed across an enterprise and locally enforced using a host Access Control List in [6]. An implementation of NGAC as Next-generation Database Access control [10], NDAC, provides a means of expressing policies over (Structured Query Language) SQL queries for accessing data in tables, rows, and columns in existing (Relational Data Stream Management System) RDMS products.

Currently, the NGAC reference model only supports users' capabilities and access entries audits. These existing policy review features of the PM were improved upon by reducing the overall time complexity from cubic to linear run time [3]. Also, a faster theoretical approach for both per-user and per-object policy review was proposed [2]. Beyond the application and the improvement of existing features of the policy machine, we addressed one of the vital administrative review queries that the PM cannot currently answer in this work.

Rather than the traditional rule-based ABAC policy specification approach, the policy machine has a unique and simplified approach that implies policy specification through the RBAC-style relations. Thus, policy review is in polynomial time [17]. In ABAC models like [15], [16], policies are in propositional logic. Consequently, police review is an NP-complete or even undecidable problem. Since PM specifies access through relations enumerated between attributes, relations among attributes of the same type are hierarchical. One benefit of the successive ranks among attributes is granting access in multiple approaches. This characteristic can enhances access control flexibility and facilitates attribute management and administration. However, there is no established process to provide all the multiple ways of granting access in the policy machine implementation.

In summary, the contribution of this work is:

- The development of an algorithm that generates all the possible relations that can grant access for a denied administrative access request. In scenarios where creating an easy to see relations may be unacceptable, our algorithm can provide the PM administrator with subtle approaches.

The reminder of this paper is structured as follows. In Section II, we touch on related work on this subject. Section III provides overview of the policy machine framework and its components. Section IV presents the scope, problem statement and observation. Section V describe the approaches to grant administrative access and presents its algorithm. An

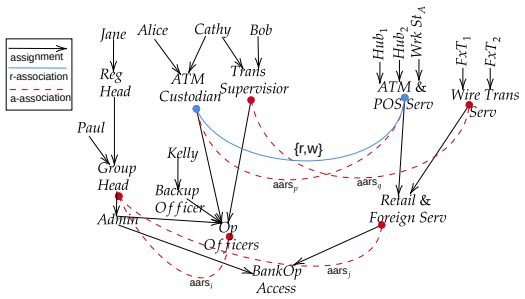


Fig. 1: Policy Machine Authorization Graph

implementation of our algorithm is in Section VI, and Section VII conclusions this work.

## II. Related Work

In NIST RBAC [5], the administrative review features define requirements in terms of an administrative interface and an associated set of semantics that provide the capability to perform policy query on RBAC elements and relations. Some of the queries answered by the administrative review functions return the set of users assigned to a given role, roles assigned to a given user, operations a given user may perform on a given object, all permissions either directly granted to or inherited by a given role, and roles directly assigned to a given user as well as those "roles that were inherited by the directly assigned roles." Fernandez et al. [9] provides axiomatic and operational semantics for ABAC policies and evaluates review queries such as permissions associated with a given principal, set of principals authorized to perform a given operation on a resource, and categories principals and resources belong. The NIST Policy Machine specification captures the policy review questions answered in C-ABAC [9], and the only difference is terminology.

Efforts related to policy review in the Policy Machine attempt to improve the efficiency of answering two types of user queries. One, can a particular user operate on an object? Two, what privileges does a user have? Mell et al. [3] contribution improves on the time complexity of performing the mentioned queries from cubic to linear time, utilizing the breadth-first search (BFS) and depth-first search (DFS) variants.

In contrast, we proposed an algorithm for the policy review question that neither the policy machine specification nor other contributions answered. Our objective is not to improve on an existing policy review question that the policy machine can answer, our algorithm extends the policy review capabilities of the PM. As the policy machine is different from the traditional rule-based ABAC policy specification approach, a query for all possible approaches to grant access is equally important as those previously explored queries.

## III. Policy Machine Overview

The rest of this section provides an overview of policy machine core data elements and selected relations pertinent to this work. The assignment, association, prohibition, and obligation are the primary relations. Privileges and restrictions are derived relations from associations and prohibitions in the PM respectively. In this work, we only focus on the assignment and association relations and concepts related to them.

### A. PM Basic Elements And Relations

The authorization (access control) state of the policy machine is an annotated Directed Acyclic Graph (DAG). Basic elements of the PM called Policy Elements (PE) are the nodes in the access control graph. These nodes are the finite sets of Users (U), Objects (O), User Attributes (UA), Object Attributes (OA), and Policy Classes (PC). As shown in the access control graph of figure 1, the users are the left top nodes. Unlabeled directed acyclic edges called assignments are relations allowed from user nodes to user attribute nodes, user attribute to user attribute nodes, and user attribute to policy class node. In a similar fashion on the right side of the graph, connected unlabeled directed acyclic edges start from object nodes, through objects attribute nodes, and terminates at the policy class node.

All the nodes except for the policy class must have a path to the policy class. Users' access to protected resources is only possible through the creation of an association. An association is a relation represented by labeled (annotated) downward-arcing edge from a user attribute node to an attribute (user attribute or object attribute node). For instance, in figure 1, the association triple (*Group Head*, $aars_i$, *Retail & Foreign Serv*) specify that a user who has a path to *Group Head* is authorized to perform operations enabled by $aars_i$ on *Retail & Foreign Serv* and policy element that has a path to *Retail & Foreign Serv*. Access granted through an association could be a set of resource access rights (i.e., r-association in the legend) or a set of administrative access rights (i.e., a-association in the legend) shown in solid blue and dashed red edges respectively. The policy elements and the relations constitute the authorization graph.

## IV. Scope, Problem Statement, and Observation

In the context of policy review, previous contributions were to improve on an existing algorithm in the policy machine [2], [3]. However, as we have mentioned before, there are other policy review problems that are not addressed in the policy machine framework. In the following subsection we specify the aspects of the PM framework covered in this work and we present an example to illustrate the problem we have proposed an algorithm to solve.

### A. Scope

The PM framework has multiple facets – the policy elements and the assignments that make up a policy element diagram, the association and prohibition that apply the policy element diagram to form the authorization graph, and obligation that are carried out when access related event occur [1]. Prohibition and obligation are outside the scope of this paper, we focus only on administrative access rights granted through assignment and association.

### B. Problem Statement

Users are granted access to protected resources in policy machine by creating assignments (unlabeled edges) and/or association (labeled edges) between conforming policy elements (nodes). Based on the hierarchical nature of the policy machine authorization graph, access for a specific request is granted in multiple ways. In other words, multiple paths can be created from a user or user attribute to a user attribute granted access rights over some policy elements. However, for various reasons, it may happen that not all the obvious ways of granting an access are in accordance with specified policy. With the algorithm we proposed in this paper, an exhaustive list of all possible approaches to authorize a request is generated. The output of our algorithm can help an administrator to decide an approach to authorize an access.

The example that follows demonstrates how the number of approaches to grant access can explode and how any constraint can limit the number of ways an access may be granted.

**Example 1.** The authorization graph of figure 1 represents the access policy of a financial institution with the policy class *BankOp Access*. In the financial institution, a sensitive task 'trans-T' must be completed by a single user who is a member of both user attributes, *ATM Custodian* and *Trans Serv Supervision* (i.e., *Cathy*). The task 'trans-T' is modeled as a sequence of administrative operation granted on Object attributes, *ATM & POS Serv* and *Wire Trans Serv* through the sets of administrative access rights $aars_p$ and $aars_q$ respectively. *Alice* and *Bob* are employees of the financial institution. Alice is a member of *ATM Custodian* and not *Trans Serv Supervision*, while Bob is a member of *Trans Serv Supervision* and not *ATM Custodian*.

For another task 'T-1' in this same institution, it is required that a user who is a member of both *ATM Custodian* and *Trans Serv Supervision* assigns a member of the *Backup Officer* to *ATM Custodian* in order to complete the task 'T-1'. In the current state of the authorization graph of figure 1, *Cathy* does not have the access right to assign *Backup Officer* to *ATM Custodian*. Assuming *Jane* and *Paul* (members of the institution that has a path to the user attribute *Group Head*) can grant *Cathy* the access right to create the required assignment (unlabeled edge), using the administrative access rights in $aars_i$. The following are approaches to grant *Cathy* the access right to assign *Backup Officer* to *ATM Custodian*:

1) *Create association between user attribute: Jane* or *Paul* can create associations from user attributes that are descendant nodes to $Cathy$ to *Backup Officer*, and its user attributes descendants. That is, the predecessor and successor nodes of an association that grants $Cathy$'s request are the sets *ATM Custodian*, *Trans Serv Supervision*, *Op Officers*, and *Backup Officer*, *Op Officers* respectfully. Assuming $aars_k$ is an administrative access right set that enables $Cathy$'s request. Using association only, any of the triples in the set, { (*ATM Custodian*, $aars_k$, *Backup Officer*), (*ATM Custodian*, $aars_k$, *Op Officers*), ( *Trans Serv Supervision*, $aars_k$, *Backup Officer*), (*Trans Serv Supervision*, $aars_k$, *Op Officers*), (*Op Officers*, $aars_k$, *Backup Officer*), (*Op Officers*, $aars_k$, (*Op Officers*) } authorizes Cathy's request.
2) *Create assignment from a user attribute to user attribute:* An existing association (*Group Head*, $aars_i$, *Op Officers*) grants *Op Officers* and its ancestor nodes the authority to perform operations enabled by $aars_i$ (that includes $Cathy$'s request) on *Op Officers* and its ancestor nodes. *Jane* or *Paul* can create an assignment from DAG conforming user attributes that are descendants of $Cathy$ to *Group Head* and its user attribute ancestor nodes. The cartesian product of the sets {*ATM Custodian*, *Trans Serv Supervision*}, and {*Group Head*, *Regional Head*} are the user attribute to user attribute relations that authorize $Cathy$'s request.
3) *Create assignment from a user to user attribute:* Again, *Jane* or *Paul* can create assignments (unlabeled edges) from user node (*Cathy*) to user attribute node (*Group Head* or *Regional Head*) that makes (*Group Head* or *Regional Head*) reachable from *Cathy*. There are only two ways to authorize *Cathy*'s request using this approach.

In total, there are twelve different ways of creating an assignment or association to allow *Cathy* complete task 'T-1'. However, only two out of the twelve possible relations does not violate the constraint on the task 'trans-T'. If *Jane* or *Paul* should authorize *Cathy*'s request using the approaches enumerated in (1) and (2), then *Alice* and *Bob* can collude to carry out task 'trans-T'.

Even more, this is a simple example compare to the size of an enterprise where this kind of issue get more complicated. Also, the structure of an authorization graph may permit granting an access using any non-redundant combination of the three approaches. For instance, in the previous example, *Jane* or *Paul* may grant *Cathy*'s request to complete task 'T-1' by first creating an (unlabeled edge) assignment from *Cathy* to *Backup Officer* and then create another (unlabeled edge) assignment from *Backup Officer* to *Group Head* or *Regional Head*.

### C. Observation

The Principal Authority (PA), also known as the superuser, is a compulsory predefined entity of the PM. The PA is responsible for creating and controlling the policies of the PM in their entirety and inherently holds universal authorization to carry out those activities within the PM framework.

The access rights held by the principal administrator can be delegated to a domain or subordinate administrators except the access right to create and delete policy class and the access right to create and delete assignment of attributes to the policy class. For instance, the Principle Administrator (PA) that instantiated the authorization graph of figure 1 is not visible in the graph. He created the policy elements and relations as shown in the figure. Access rights granted to users with a path to *Group Head* are sufficient to create all other policy elements and relations.

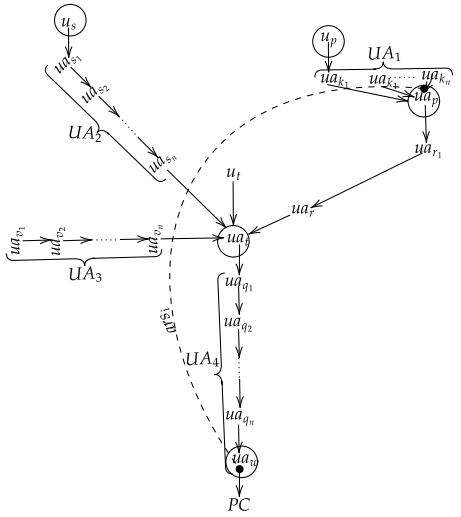


Fig. 2: A Generic Authorization Graph for Scenario-I

## V. Policy Review And Authorization

To authorize access in the PM, there are two possible scenarios. In this section, we discuss these scenarios, the policy review of policy elements involved in granting access, and how we apply these scenarios and these policy elements in our algorithm.

### A. Access Request Scenarios

Authorization approaches depends on the relationship between a user requesting access and the protected resource in an authorization graph. We interchangeably refer to the user seeking access as the *source user* and the protected resource as the *target policy element*. There are two types of relationships between the *source user* and the *target policy element*, viz:

1) There is a path between the *source user* and the *target policy element*. This can only happen if the *target policy element* is a user or a user attribute.
2) There is no path between the *source user* and the *target policy element*. In this scenario, the *target policy element* is of a user, user attribute, an object, or an object attribute type.

Our algorithm only considers granting access by a non-principle administrator through the creation of edges (relations) between existing nodes (policy elements).

### B. Access Enablers

To authorize access, a user create one or more edges (relations) between policy elements through associations. We refer to this user as the *privileged user*, the policy elements as the *access enabling policy elements*, and the associations as the *access enabling associations*. The path relationship between other policy elements and the *source user*, *privileged user*, *access enabling associations*, or *target policy element* defines the *access enabling policy elements*.

Scenario-I requires at least one *access enabling association* to authorize a *source user*'s access. An association that has its predecessor-node reachable from a *privileged user*, its successor-node reachable from the *source user* and its label has the access right the *source user* is requesting. In other words, there is a user that can perform the administrative operation another user is requesting and his capable of granting this access.

On the other hand, scenario-II requires at the minimum two *access enabling associations*. There is an association through which the *privileged user* can perform the requested access. And a second association exists through which the *privileged user* can assign the *source user* to the former.

In both scenarios, we adapt following notations for specifying elements, functions, and sets in the *access enabling policy elements*

- $\texttt{tail} : \texttt{ASSOCIATION} \longrightarrow \texttt{UA}$: is a function that maps an edge, association relation, $(ua_i, ars_j, at_k) \in \texttt{ASSOCIATION}$ to the (user attribute) node $ua_i \in \texttt{UA}$ it originates.
- $\texttt{head} : \texttt{ASSOCIATION} \longrightarrow \texttt{AT}$: is a function that maps an edge, association relation, $(ua_i, ars_j, at_k) \in \texttt{ASSOCIATION}$ to the (user/object attribute) node $at_k \in \texttt{AT}$ it terminates. Where $\texttt{AT} = \texttt{UA} \cup \texttt{OA}$
- $\texttt{anc}: \texttt{PE} \longrightarrow 2^{\texttt{PE}}$: is the mapping from a policy element to the set of policy elements that is an ancestor to the policy element.
- $\texttt{des}: \texttt{PE} \longrightarrow 2^{\texttt{PE}}$: is the mapping from a policy element to the set of policy elements that is a descendant to the policy element.
- $\texttt{PE}_{i_{\texttt{func}}} = \{\texttt{node} \mid (\exists pe_j \in \texttt{PE}_i)[\texttt{node} \in \texttt{func}(pe_j)]\}$: is the set of all policy elements returned by $\texttt{func}$ for the set $\texttt{PE}_i$, where $\texttt{func}$ is $\texttt{anc}$ or $\texttt{des}$.

*1) Scenario-I Access Enabling Policy Elements:* Figure 2 illustrates a generic authorization graph for this scenario. Assuming the *source user* ($u_s$) is requesting an access right that is a subset of $\texttt{ars}_i$, the label of *access enabling association* $(\texttt{ua}_p, \texttt{ars}_i, \texttt{ua}_w)$ on target ($\texttt{ua}_t$), the following are the *access enabling policy elements* sets.

i) $UA_1 = \{\texttt{ua} \mid \texttt{ua} = \texttt{tail}((\texttt{ua}_p, \texttt{ars}_i, \texttt{ua}_w)) \vee \texttt{ua} \in \texttt{anc}(\texttt{tail}((\texttt{ua}_p, \texttt{ars}_i, \texttt{ua}_w))\}$
ii) $UA_2 = \{\texttt{ua} \mid \texttt{ua} \in \texttt{anc}(\texttt{head}((\texttt{ua}_p, \texttt{ars}_i, \texttt{ua}_w))) \wedge \texttt{ua} \in \texttt{des}(u_s)\}$
iii) $UA_3 = \{\texttt{ua} \mid \texttt{ua} \in \texttt{anc}(\texttt{head}((\texttt{ua}_p, \texttt{ars}_i, \texttt{ua}_w))), \texttt{ua} \notin UA_{2_{\texttt{anc}}}, \texttt{ua} \notin UA_2, \texttt{ua} \notin UA_{1_{\texttt{des}}}, \texttt{ua} \notin UA_1\}$
iv) $UA_4 = \{\texttt{ua} \mid \texttt{ua} \in \texttt{des}(u_s) \wedge \texttt{des}(\texttt{ua}_t)\}$

*2) Scenario-II Access Enabling Policy Elements:* An object attribute $\texttt{oa}_w$ is the *target policy element* of figure 3. Let the association granting the *privileged user* $u_p$ the access right to perform the *source user*'s request be *access enabling association'* $(\texttt{ua}_i, \texttt{ars}_q, \texttt{oa}_r)$ and the association the *privileged user* can use to create a path from the *source user* $u_s$ to the predecessor-node of *access enabling association'* be *access enabling association''* $(\texttt{ua}_i, \texttt{ars}_j, \texttt{ua}_k)$. The definition for the user attribute *access enabling policy elements* sets $UA_1$,

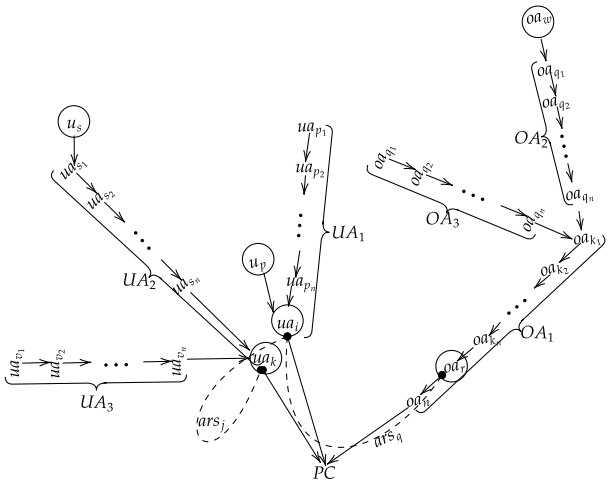


Fig. 3: A Generic Authorization Graph for Scenario-II

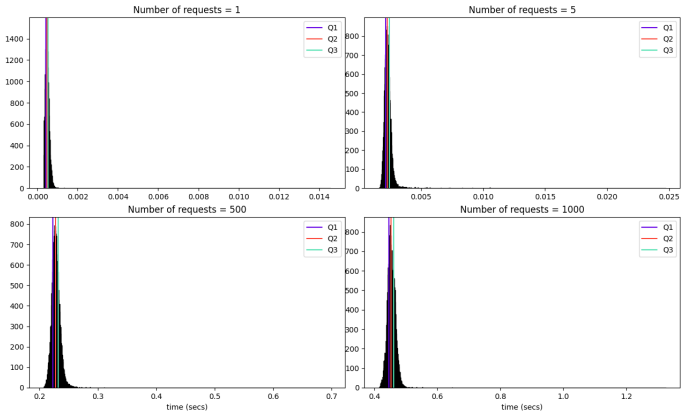


Fig. 4: Distribution of Response Time to Generate Access Authorization Approaches for Scenario-I on graphs with 1,000 nodes

$UA_2$, and $UA_3$ in the previous scenario is the same here. In addition, the object attribute *access enabling policy elements* sets are:

i) $OA_1$ = {$oa$ | $oa \in \texttt{anc}(\texttt{head}((\texttt{ua}_i, \texttt{ars}_q, \texttt{oa}_r)))$, $oa \notin \texttt{des}(\texttt{oa}_w)$, $oa \neq \texttt{oa}_w$}

ii) $OA_2$ = {$oa$ | $(oa \in \texttt{des}(\texttt{oa}_w) \wedge oa \notin OA_{1_{\texttt{des}}}) \vee oa = \texttt{oa}_w$}

iii) $OA_3$ = {$oa$ | $(oa \in \texttt{anc}(\texttt{head}((\texttt{ua}_i, \texttt{ars}_q, \texttt{oa}_r)))$, $oa \in \texttt{des}(\texttt{oa}_w)$, $oa \in OA_{2_{\texttt{des}}}) \vee oa = \texttt{head}((\texttt{ua}_i, \texttt{ars}_q, \texttt{oa}_r))$ }

### C. Algorithm And Authorization Approaches

An input to the main function `mainPReview` is the authorization state of a graph $G_{auth}$ and access request triple (i.e., the *source user*, the access right *source user* is requesting, and the *target policy element*). The expected output of this algorithm is a list of sets of possible approaches to grant the *source user*'s access. The `mainPReview` function uses the access request triple to determine the applicable scenario and queries for the *access enabling association*(s).

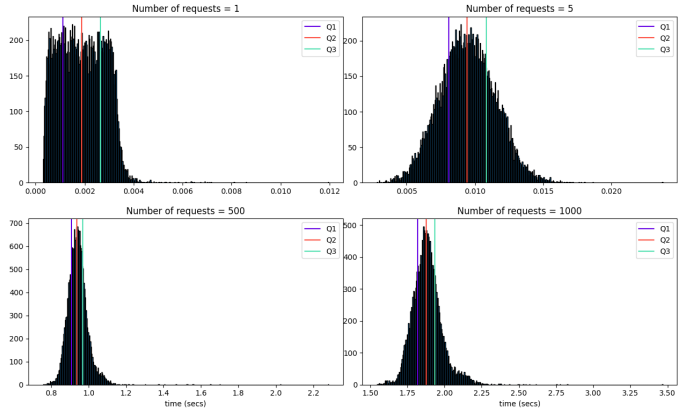


Fig. 5: Distribution of Response Time to Generate Access Authorization Approaches for Scenario-II on graphs with 1,000 nodes

The functions `scenario-IPEGenerator` and `scenario-IIPEGenerator` take the *access enabling association*(s) as input parameters to return the sets of *privileged user* and *access enabling policy elements* for the respective scenarios. The *privileged user*s' access rights are utilized to create the authorization approaches. All administrative access rights for creating assignments enable a common operation, `createAssign`. The `createAssign` takes an ordered pair of nodes that an edge originates and terminates as input. Similarly, access rights for creating associations enable the operation `createAssoc`. A triple as input to the `createAssoc` is the predecessor-node, the label, and successor-node of an edge.

Another pair of functions, `sc-IAuthGen`, and `sc-IIAuthGen` use the power set of access right set granted to the *privileged user* to request the creation of relations among the *access enabling policy elements* for scenario-I and scenario-II respectively.

TABLE I: Proportion of node types for 100 random graphs with 1,000 nodes

| **Node types** | ***Scenario-I*** | ***Scenario-II*** |
|---|---|---|
| Users | 400 | 200 |
| Objects | - | 200 |
| User attributes | 600 | 300 |
| Object attributes | - | 300 |
| Policy class | 1 | 1 |

## VI. Implementation

In this section, we discuss the implementation details of our algorithm utilizing the networkx (python library for studying graphs and networks). An Ubuntu virtual machine with two cores and 10Gb of memory was used as our experimental platform. We tested the response time to return all the approaches that authorize request(s) by simulating 100 random

graphs for the two scenarios. For each of the random graphs, we simulated 200 variations for each request sizes of 1, 5, 500, and 1000. The 100 random access control graphs have 1000 (user, user attribute, object, and object attribute) nodes.

The proportion of nodes of each type per access control graph is shown in the table I. We restricted the maximum number of edges from a user or object to the policy class node to 5. We did this by grouping user and object attributes into four levels (labeled 1 to 4) while the policy class is labeled level 0. Edges (assignment relations) were only allowed from user and object attributes at higher levels to user and object attributes at lower levels. That is, no assignment relations between attributes at the same level.

In the random authorization graphs generated for scenario-I, association relation predecessor nodes are reachable from a *privileged user*. Also, the successor nodes of the association relations are reachable from *target policy element* (user attribute) and *source user*. For each request in the scenario-II random authorization graphs, there are at least two association relations. One of the association relations has its successor node reachable from *target policy element* (object or object attribute), while the other enables a *privileged user* to delegate access right to *source user*.

Figure 4 and 5 are the distribution of response time to generate all possible approaches to grant a request in the two scenarios. The figures are annotated with the quartile response time, ($Q_1$, $Q_2$, and $Q_3$) as shown in the figures' legend. In scenario-I, the response time for 75% of access request sizes of 1, 5, 500, and 1000 is below 0.53 x $10^{-3}$, 2.25 x $10^{-3}$, 0.25, and 0.46 seconds, respectively. For scenario-II, the response time for 75% of access request sizes of 1, 5, 500, and 1000 is below 2.6 x $10^{-3}$, 11 x $10^{-3}$, 0.97, and 1.93 seconds, respectively. The distribution of all the plots for the response time is right-skewed. Few instances that the response time takes a longer time, the cardinality of all the sets of access enabling policy elements are non-empty. For cases that the response time takes a short time, some of the policy element sets are empty. Again, the NIST policy machine specification, and no related effort answer this policy query of our work, no comparative experiment for us to perform.

## VII. Conclusion And Future Work

The policy machine attribute-based access control model uses enumeration of attributes and relations to formulate policy. This feature makes performing (queries) reviews of policy feasible, rather than the NP-complete time complexity in models that express policy logically. However, the NIST reference specification is lacking policy reviews pertinent to the creation and modification of administrative access policy. An example in section IV-B demonstrates how policy review using our proposed algorithm can prevent an unintended consequence in the PM administration. In section VI, the response time of the performed experiments indicates the proposed algorithm is scalable. Our future work shall expand the algorithm in this paper to handle policy review of revocation and authorization with constraint.

## References


[1] D. Ferraiolo, S. Gavrila, and W. Jansen, "Policy Machine: features, architecture, and specification," National Institute of Standards and Technology, Internal Report 7987

[2] R. Basnet, S. Mukherjee, V. M. Pagadala and I. Ray, "An efficient implementation of next generation access control for the mobile health cloud," 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC), Barcelona, 2018, pp. 131-138.

[3] Peter Mell, James M. Shook, and Serban Gavrila. 2016. Restricting Insider Access Through Efficient Implementation of Multi-Policy Access Control Systems. In Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats (MIST '16). ACM, New York, NY, USA, 13-22.

[4] Smriti Bhatt, Farhan Patwa, and Ravi Sandhu. 2017. ABAC with Group Attributes and Attribute Hierarchies Utilizing the Policy Machine. In Proceedings of the 2nd ACM Workshop on Attribute-Based Access Control (ABAC '17). ACM, New York, NY, USA, 17-28. DOI:

[5] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," ACM Transactions on Information and System Security (TISSEC), vol. 4, no. 3, pp. 224-274, 2001.

[6] David Ferraiolo, Serban Gavrila, and Gopi Katwala. 2018. A System for Centralized ABAC Policy Administration and Local ABAC Policy Decision and Enforcement in Host Systems using Access Control Lists. In Proceedings of the Third ACM Workshop on Attribute-Based Access Control (ABAC'18). ACM, New York, NY, USA, 35-42.

[7] Bruhadeshwar Bezawada, Kyle Haefner, and Indrakshi Ray. 2018. Securing Home IoT Environments with Attribute-Based Access Control. In Proceedings of the Third ACM Workshop on Attribute-Based Access Control (ABAC'18). ACM, New York, NY, USA, 43-53.

[8] K. K. Kolluru, C. Paniagua, J. van Deventer, J. Eliasson, J. Delsing and R. J. DeLong, "An AAA solution for securing industrial IoT devices using next generation access control," 2018 IEEE Industrial Cyber-Physical Systems (ICPS), St. Petersburg, 2018, pp. 737-742.

[9] Maribel Fernandez, Ian Mackie, and Bhavani Thuraisingham. 2019. Specification and Analysis of ABAC Policies via the Category-based Metamodel. In Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy (CODASPY '19). Association for Computing Machinery, New York, NY, USA, 173-184.

[10] David Ferraiolo, Serban Gavrila, Gopi Katwala, and Joshua Roberts. 2017. Imposing Fine-grain Next Generation Access Control over Database Queries. In Proceedings of the 2nd ACM Workshop on Attribute-Based Access Control (ABAC '17). ACM, New York, NY, USA, 9-15.

[11] S. Bhatt, F. Patwa and R. Sandhu, "An Attribute-Based Access Control Extension for OpenStack and Its Enforcement Utilizing the Policy Machine," 2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC), Pittsburgh, PA, 2016, pp. 37-45.

[12] Pagadala V., Ray I. (2019) Achieving Mobile-Health Privacy Using Attribute-Based Access Control. In: Zincir-Heywood N., Bonfante G., Debbabi M., Garcia-Alfaro J. (eds) Foundations and Practice of Security. FPS 2018. Lecture Notes in Computer Science, vol 11358. Springer, Cham.

[13] 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.

[14] 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.

[15] X. Jin, R. Krishnan, and R. S. Sandhu. A unified attribute-based access control model covering DAC, MAC and RBAC. DBSec 2012: Data and Applications Security and Privacy XXVI. Springer, Berlin, Heidelberg

[16] D. Servos and S. L. Osborn. HGABAC: Towards a formal model of hierarchical attribute-based access control. Foundations and Practice of Security FPS 2014. Springer Cham

[17] P. Biswas, R. Sandhu, and R. Krishnan. Label-based access control: An ABAC model with enumerated authorization policy. Proc of the 2016 ACM International Workshop on Attribute Based Access Control. ACM Press.