IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 31 May 2019; revised 13 September 2019; accepted 19 September 2019.
Date of publication 1 October 2019; date of current version 6 December 2021.

Digital Object Identifier 10.1109/TETC.2019.2944787

Enabling Workforce Optimization in Constrained Attribute-Based Access Control Systems

ARINDAM ROY, SHAMIK SURAL, (Senior Member, IEEE), ARUN KUMAR MAJUMDAR, (Senior Member, IEEE), JAIDEEP VAIDYA, (Senior Member, IEEE), AND VIJAYALAKSHMI ATLURI, (Senior Member, IEEE)

A. Roy is with Goa Institute of Management, Goa 403505, India
S. Sural and A. K. Majumdar (retired) are with the Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur 721302, India
V. Atluri and J. Vaidya are with the MSIS Department, Rutgers University, New Brunswick NJ 07102, USA
CORRESPONDING AUTHOR: J. VAIDYA (jsvaidya@business.rutgers.edu)

ABSTRACT Effective utilization of human capital is one of the key requirements for any successful business endeavor, with reorganization necessary if there are nonproductive employees or employees that are retiring. However, while reorganizing tasks for newer employees, it should be ensured that the employees have the requisite capabilities of handling the assigned tasks. Furthermore, security constraints forbid any arbitrary assignment of tasks to employees and also enforce major dependencies on other employees who have access to the same tasks. Since Attribute Based Access Control (ABAC) is poised to emerge as the *de facto* model for specifying access control policies in commercial information systems, we consider organizational policies and constraints to be modeled with ABAC. Given the increasing size and scale of organizations, both in terms of employees and resources that need to be managed, it is crucial that computational solutions are developed to automate the process of employee to task assignment. In this work, we define the Employee Replacement Problem (ERP) which answers the question of whether a given set of employees can be replaced by a smaller set of employees, while ensuring that the desired security constraints are not violated. We prove that the problem is NP-hard and use CNF-SAT to obtain a solution. An extensive experimental evaluation is carried out on diverse data sets to validate the efficiency of the proposed solution.

INDEX TERMS ABAC, workforce optimization, downsizing, separation of duty, binding of duty

I. INTRODUCTION

Business information systems use different kinds of access control mechanisms in order to prevent the risk of unauthorized access to their resources. The underlying models for such mechanisms have undergone several generations of development over the last few decades. Before the advent of Role Based Access Control (RBAC) in the early 1990s, the prevalent access control models were Discretionary Access Control (DAC) and Mandatory Access Control (MAC). In DAC, the owner of an object directly controls the assignment and transfer of rights associated with the objects she owns [25]. The term discretionary is used since it is at the discretion of the owner whether to confer rights to access an object to others. In MAC, access is restricted on the basis of the sensitivity level of objects and the security clearance of subjects [5]. On the other hand, RBAC grants access to objects based on the roles assigned to its users. [35]. The effectiveness of various access control models in terms of managing access to workflow systems, which divide a job into a set of well-defined tasks that need to be executed in a pre-specified sequence, has also been studied [42].

Although RBAC has been quite popular and is still being used in fairly diverse commercial information systems [13], [44], [45], it lacks extendibility and has certain other limitations. For example, RBAC is not suitable for systems where access decisions may have to be made without prior knowledge of the subjects and they could also depend on the context in which an access request is made. For instance, an access control policy in such a system could depend on the designation of the user, type of the object, and access request location and time. In a situation where a large number of objects exist, an RBAC system can also face the problem of permission explosion since the permissions are specified in terms of individual objects.

To overcome such limitations of RBAC, the Attribute Based Access Control (ABAC) model [19] has recently been proposed. When a subject in an ABAC system requests to perform certain operations on an object, the request is granted or denied based on the assigned attribute values of the subject and the object, the environmental conditions as well as a set of rules that are specified in terms of those attribute values and conditions. DAC, MAC and RBAC can be viewed as special cases of ABAC. Due to its ability to express different types of access control policies including fine-grained access control, ABAC is expected to encompass various domains including databases, operating systems, networks and applications. It is also expected to emerge as the *de facto* paradigm for the specification and enforcement of access control policies [14], [26]. In terms of the attributes used, DAC works on the basis of the "identity" attribute, MAC works on the basis of the "clearance" and "sensitivity" attributes, whereas RBAC works on the basis of the "role" attribute. Several aspects of effective deployment of ABAC in various commercial systems have been looked into in the access control literature [3], [11], [17], [18], [43].

It may be noted that access control models support various types of constraints necessary to enforce different security policies in a computer system. Two of the most important and widely used constraints are Separation of Duty (SoD) and Binding of Duty (BoD) constraints. An SoD constraint requires that a set of tasks should not be carried out by a single user. For example, in a banking application, issuing a check and authorizing it should not be done by the same staff member. On the other hand, a BoD constraint requires that if a particular employee is handling a particular task, she will have to handle some other specified tasks too. For example, only a full professor can be the chair of a department. Enforcement of such constraints reduces the potential for fraud, and are often dictated by the stakeholders. Other than SoD and BoD, often model specific security constraints are also used. For example, RBAC supports Cardinality Constraint, Prerequisite Constraint [35], Statically Mutually Exclusive Roles (SMER) Constraint [23], etc. Further, there can be constraints which enforce the assignment of roles to users based on their capabilities.

Likewise in ABAC, other than SoD and BoD, an organization often requires specifying a prerequisite, which dictates that a particular user can only be assigned to a particular attribute if she is already assigned to some other specified attributes. For example, a user can have her *clearance level* as *high* only if the value of the *designation* attribute is *Director*. Moreover, in an ABAC system, each user can be specified with the attribute values that she is capable of holding. In such cases, the organization would be able to assign tasks to a user only according to her capabilities, which is intuitively obvious. Similar to RBAC, an ABAC system can also support cardinality constraint, which allows only a particular maximum number of employees to be assigned to a particular attribute value.

For the success of any business organization irrespective of the access control model it deploys, one of the key requirements is the ability to efficiently manage its human capital. Assigning the right person to the right position at the right time and location is imperative for its management to function effectively. A critical task towards maximizing human resource utilization involves replacing non-performing employees with more productive substitutes if the need arises. Moreover, if a number of employees leave the organization looking for other opportunities or on retirement, filling up the vacant positions with new candidates provides an opportunity to improve the overall manpower deployment matrix. However, such assignments are often constrained by the various access control policies prevalent in the organization.

With an objective to optimize the workforce, organizations, including the ones that have deployed any specific access control model and security constraints, need to reorganize their workforce from time to time. As a cost-effective measure, it is desirable to replace a set of retiring employees with a smaller number of new employees [1]. Additionally, under-productive employees often need to be replaced with a smaller group of new employees. While this could be done manually, given the increasing size and scale of organizations and the diversity of resources (both internal and external, such as on the cloud), it is crucial to develop computational solutions to automate the process of employee to task assignment. While recent work enables secure and flexible access control for users in cloud storage [22], [40], more work needs to be done to automate reassignment of employees to tasks. In this work, we address the problem of enabling such workforce optimization by replacing a subset of employees with a smaller set of employees in an organization that uses ABAC as the access control model.

The rest of the paper is organized as follows. In Section II we present a formal definition of ABAC and also precisely define the various types of constraints in ABAC. Section III introduces the Employee Replacement Problem and explains the same using an illustrative example. The complexity class of the problem is studied in Section IV and our proposed solution to the problem is discussed in Section V. Section VI presents the results of the experimental evaluation. Previous works related to the current study are discussed in Section VII. Finally, Section VIII draws conclusions and discusses plans for future research.

II. ATTRIBUTE BASED ACCESS CONTROL (ABAC)

ABAC is an access control model in which access rights are granted to users through the use of authorization policies based on the notion of attributes of the requesting user, requested object and the environment in which a request is made. ABAC is comprised of the following basic components [19]:

- A set U of users and a set U_A of user attributes. A member of U is represented as u_i, for $1 \leq \mathrm{i} \leq |U|$ and that of U_A as ua_j, for $1 \leq \mathrm{j} \leq |U_A|$. Each attribute ua_j can acquire a value from an associated set of possible values. For example, consider that a user attribute *Marital Status* is associated with the set of values {*single*, *married*, *separated*, *divorced*, *widowed*, *Null*}, where *Null* indicates that the value of the attribute is unknown.

Here, each user $u_i \in U$ can have her attribute *Marital Status* assigned to any of the possible values.

- A set O of sensitive objects and a set O_A of object attributes. A member of O is represented as o_i, for $1 \leq i \leq |O|$ and that of O_A as oa_j, for $1 \leq j \leq |O_A|$. Similar to users, each attribute oa_j can acquire a value from an associated set of possible values. For instance, consider the set {*Audio*, *Video*, *Text*} to be associated to an object attribute *Type*. Then, each object $o_i \in O$ can have its attribute *Type* assigned to *Audio*, *Video* or *Text*.
- A set E of environment specifications which are independent of users and objects and a set E_A of environment attributes. Spatial or temporal conditions are examples of such specifications. A member of E is represented as e_i, for $1 \leq i \leq |E|$ and that of E_A as ea_j, for $1 \leq j \leq |E_A|$. Each element in E_A is assigned a value from a set associated with it. For instance, the attribute *Branch* can acquire values from the set {*Seattle*, *Dallas*, *Boston*} associated with it.
- F_U: $U \times U_A \rightarrow \{v_j | v_j$ is a user attribute value $\}$. For example, if an employee *Patrick* is married, F_U(*Patrick*, *Marital Status*) = *married*.
- F_O: $O \times O_A \rightarrow \{v_j | v_j$ is an object attribute value $\}$. As an instance, for an *Audio* file f_i, $F_O(f_i, Type)$ = *Audio*.
- F_E: $E \times E_A \rightarrow \{v_j | v_j$ is an environment attribute value $\}$. For instance, an environment e_i having its attribute *Branch* as *Seattle*can be represented as $F_E(e_i, Branch)$ = *Seattle*.
- A set A consisting of all possible actions (operations) on objects allowed in the system. For instance, if the only possible actions on a file are *append* and *delete*, then A = {*append*, *delete*}. Each member of A is represented as a_i.
- A set P of authorization rules (interchangeably called policies) which consists of members represented by pa_i, where $1 \leq i \leq |P|$. Each policy is represented by a 4 - tuple of the form $\langle uc, oc, ec, a \rangle$ [20]. Here, user conditions, object conditions and environment conditions are represented by *uc*, *oc* and *ec*, respectively. A user condition comprises equalities of the form $n = c$, where n is a user attribute name and c is either a user attribute value or *any*(similar representation is used for object and environment conditions oc and ec). The attribute with the name n becomes irrelevant for making access decisions if the value of c is *any*. $a \in A$ is the action which could be performed on the objects with object condition *oc*.

When a user makes a request to access an object, the set P of authorization policies is searched. If a policy through which the access can be granted is found, then access is granted, otherwise denied. For example, suppose an organization has a condition that only a user with designation *System Manager* and project type as *Telecom*, can read any database of the company *British Telecom* from his office computer. In ABAC, this requirement can be specified in the form of a policy $\langle$ {(Project Type = *Telecom*), (designation = *System Manager*)}, {(type = *Database*), (Company = *British Telecom*)}, {(access location = *Office*)}, *read*$\rangle$.

A permission in any access control model essentially provides authority to perform operations (actions) on certain types of objects. Hence, in an ABAC system, a permission p is defined as a pair (a, oc), where a denotes an action to be performed and oc denotes an object condition, the set of all such permissions is denoted by *Per*. The action a could be performed on the objects with object condition oc.

An SoD constraint in ABAC enforces involvement of at least a given number of users to complete a task that requires a set of permissions. On the other hand, a BoD constraint binds a user to acquire all the permissions in a set in case she requires any one of them. Using the previously articulated notion of a permission, we can define SoD and BoD in ABAC as follows.

Definition 1 (Separation of Duty Constraint): An SoD constraint in an ABAC system, denoted by $\langle PR, k \rangle$ *where* $PR \subseteq Per$ *is a set of permissions, and k is an integer, is said to be satisfied if at least k users are required to perform a task that requires all the permissions in the set PR.*

Definition 2. Binding of Duty Constraint. A BoD constraint in an ABAC system, denoted by a set of permissions $\{p_1, ..., p_n\}$*, is said to be satisfied if a user gets access to all or none of the permissions in the set through the authorization policies.*

We use the notion of an attribute-value pair (a, v) in this work, where a denotes a user attribute name and v its value. The set consisting of all possible attribute-value pairs is denoted as AV and each attribute-value pair is denoted as av_{ij}. An attribute-value pair $(a_i, v_j) \in AV$ denoted by av_{ij} is said to be assigned to a user u, if the user attribute a_i of u acquires the value v_j, i.e., $F_U(u, a_i) = v_j$. The many-to-many user to attribute-value pair assignment relation is denoted by a matrix $UA \subseteq U \times AV$. $UA(u, av) = 1$, if the user u is assigned to the attribute-value pair av and 0, otherwise. Now, we use this notation to define the User Capability UC constraint in an ABAC system.

Definition 3: User Capability (UC) Constraint. A matrix UC of size $|U| \times |AV|$ *represents a set of user capability constraints, where an entry of '1' in any* (u, av_{ij}) *assignment denotes the capability of the user u of acquiring the attribute-value pair* av_{ij}*. Likewise, a '0' entry denotes that the user u is not capable of getting assigned to the attribute-value pair* av_{ij}*.*

A set $X \subseteq U$ of users can be *replaced* with a smaller set Y of users with different capabilities if all the attribute value pairs assigned to the set X of users can be assigned to the set Y of users satisfying all the given constraints in an ABAC system. Formally, $\forall k, \forall i, \exists j$ such that, $UA(x_i, av_k) \Rightarrow UA(y_j, av_k)$, where $x_i \in X, y_i \in Y$ and $av_k \in AV$.

III. PROBLEM DEFINITION

We now introduce the problem of verifying whether a particular subset of users in an ABAC system can be *replaced* with a smaller set of users with different capabilities satisfying given SoD and BoD constraints. We formally define the problem as follows:

TABLE 1. *UA* matrix for the user set X for Example 1.

	av_{11}	av_{12}	av_{21}	av_{22}	av_{31}	av_{32}
u_1	1	0	0	1	1	0
u_2	0	0	0	0	1	1
u_3	0	1	1	0	0	0
u_4	0	1	0	1	0	1
u_5	1	0	1	0	0	1

Definition 4 (Employee Replacement Problem.) Given a set U of users, verify if a set X of users, where $X \subseteq U$ can be replaced *with a set Y of users, where $|Y| < |X|$, satisfying the set SC of SoD constraints, set BC of BoD constraints and the user capability constraint UC in an ABAC system.*

An ERP problem instance verifies whether a given set of users can replace a larger set of users in the ABAC system, satisfying all the given security constraints. It is to be noted that the only condition for a user to get assigned to an attribute value pair is to satisfy the given security constraints. Because none of the constraints considered in this problem are in terms of the environment attributes, the output of this problem is not dependent on the environment conditions of the ABAC system. Therefore, for this work, we ignore the environment conditions in the authorization policies and represent the policies using a static authorization matrix *AM*, which is defined as a many-to-many user condition-to-permission assignment relation, i.e., $AM \subseteq UCond \times Per$, where *UCond* is the set of all possible user conditions used in the authorization policies.

Example 1: Consider an ABAC system with 10 users so that $U = \{u_1, u_2, \ldots, u_{10}\}$, $X = \{u_1, \ldots, u_5\}$, $Y = \{un_1, un_2, un_3\}$ and $AV = \{av_{11}, av_{12}, av_{21}, av_{22}, av_{31}, av_{32}\}$. The user conditions are defined as follows: $uc_1 = \{av_{12}, av_{21}\}$, $uc_2 = \{av_{21}, av_{32}\}$, $uc_3 = \{av_{12}, av_{32}\}$ and $uc_4 = \{av_{11}, av_{31}\}$. *UA*, the user to attribute-value pair relation for set X, *UC* matrix for the set Y and the *AM* matrix are depicted in Tables 1, 2, and 3respectively. Let $SC = \{sc_1, sc_2, sc_3\}$, where $sc_1 = \langle\{p_1, p_2, p_3\}, 2\rangle$, $sc_2 = \langle\{p_2, p_3, p_4, p_5\}, 3\rangle$ and $sc_3 = \langle\{p_1, p_2\}, 2\rangle$. Also, let $BC = \{bc_1, bc_2\}$, where $bc_1 = \{p_1, p_6\}$ and $bc_2 = \{p_4, p_5\}$. For simplicity of the example, we assume that the SoD constraints do not affect the users of the set $U - X$. (The case where the users of the set $U - X$ are dependent on the specified SoD constraints is handled in Section V.) The objective is to verify if the set Y of users is sufficient to cover the assignments in *UA*. A possible UA' matrix which covers all the assignments in *UA* is presented in Table 4.

TABLE 2. *UC* matrix for the user set Y for Example 1.

	av_{11}	av_{12}	av_{21}	av_{22}	av_{31}	av_{32}
un_1	1	1	0	0	1	1
un_2	1	1	1	1	0	1
un_3	0	1	1	1	1	1

TABLE 3. *AM* matrix for Example 1.

	p_1	p_2	p_3	p_4	p_5	p_6
uc_1	1	0	1	0	0	1
uc_2	0	0	0	1	1	0
uc_3	1	0	0	0	0	1
uc_4	0	1	0	0	0	0

IV. COMPLEXITY ANALYSIS

In this section, we analyze the computational complexity of the ERP problem. We show that the ERP problem is NP-hard by developing a polynomial time reduction of the k-coloring problem to it. The k-coloring problem is a classical NP-hard problem which verifies whether a given graph can be properly colored using k different colors. Here, proper coloring refers to a constraint of no two adjacent vertices receiving the same color. Before showing the reduction we first present the formal definition of the k-coloring problem [8].

Definition 5 (k-coloring problem (KCP)): Given a graph G and a positive integer k, is there a proper coloring of G using k colors?

Theorem 1: ERP is NP-hard

Proof: The reduction procedure takes an instance of KCP with graph $G(V, E)$ and a positive integer k, and produces an instance of ERP in polynomial time as follows:

- Users: For each vertex in G, add an entry in the sets U and X of users, and for each possible color in the KCP instance, add a new user in the set Y, i.e., $U = \{u_1, ..., u_{|V|}\}, X = U, Y = \{un_1, ..., un_k\}$.
- Attribute-value pair: For each vertex, an attribute-value pair is added to the set AV, i.e, $AV = \{av_1, .., av_{|V|}\}$.
- User Condition: Each attribute-value pair is considered as a user condition in *UCond*, i.e., $\forall(a, v) \in AV$, $a = v \in UCond$.
- Permission: An entry is added to the set *Per* of permissions for each vertex in G, i.e, $Per = \{p_1, ..., p_{|V|}\}$.
- User to attribute-value pair relation (*UA*): *UA* is set as a diagonal matrix with all non-zero entries as 1. Formally, $\forall i, j$, if $i = j$, $UA(u_i, av_j) = 1$, else $UA(u_i, av_j) = 0$.
- Authorization matrix: Similarly, *AM* is set as a diagonal matrix with all non-zero entries as 1. Formally, $AM \subseteq UCond \times Per$, where $\forall i, j$, if $i = j$, $AM(uc_i, p_j) = 1$, else $AM(uc_i, p_j) = 0$.
- Constraints:
  - UC constraint: $UC(u_i, av_j) = 1$, if $\forall i, u_i \in Y$, i.e., the new users in Y are capable of having all the attribute-value pairs.

TABLE 4. An instance of the UA' matrix in Example 1.

	av_{11}	av_{12}	av_{21}	av_{22}	av_{31}	av_{32}
un_1	1	0	0	0	1	1
un_2	1	1	1	1	0	1
un_3	0	1	1	1	1	1

– SoD constraint: Each edge (v_i, v_j) denotes an SoD constraint $\langle\{p_i, p_j\}, 2\rangle$.
– BoD constraint: Each permission p_i denotes a BoD constraint $\{P_i\}$.

Now, the solution of the reduced ERP problem instance can be used to solve the corresponding KCP instance. This is proved by deducing that the following two cases hold.

– if the reduced instance of ERP is a "Yes"-instance, the corresponding KCP instance is also a "Yes"-instance.
– if the reduced instance of ERP is a "No"-instance, the corresponding KCP instance is also a "No"-instance.

Let the reduced instance of the ERP problem be a "Yes" instance. Then the set Y of k users is capable of replacing the set X of users. The user to attribute-value pair assignment relation UA' obtained after solving the ERP instance can be transformed into a color to vertex assignment relation CV for solving the corresponding KCP instance. Each user u_i in UA' corresponds to a color c_i in CV and each assignment (u_i, av_j) in UA' corresponds to the vertex v_j. It is to be noted that, since in the reduced ERP instance, the assignments in UA are such that at most one user is assigned to each attribute value pair, to construct CV, we need to consider only one assignment for each av_i from UA'. The color to vertex assignment relation CV of the KCP instance represents a proper coloring, because each edge (v_i, v_j) in the graph G will have a corresponding SoD constraint $\langle\{p_i, p_j\}, 2\rangle$ in the ERP instance.

Now, let us consider a pair α and β of instances of KCP and ERP respectively, where β is obtained from α using the proposed reduction. The second case can be written as a contrapositive statement as follows - "if α is a 'Yes'-instance, β will also be a 'Yes' instance". In that case, the proper coloring CV obtained from the KCP instance can be used to construct the UA' relation of the ERP instance. Here, the colors and vertices from CV correspond to the users and attribute value pairs in UA' respectively. The SoD constraints in the ERP instance are satisfied because they correspond to the edges in the graph G of the KCP instance. Moreover, since the vertices of the graph G also represent the assignments in UA, the fact that k colors are enough to properly color the nodes of the G implies that the set Y of k users is capable of restoring the lost assignments. □

Algorithm 1. Algorithm to Solve ERP

Input: user assignment relation UA, authorization matrix AM, user conditions $UCond$, user capability constraint UC, SoD constraints SC, BoD constraints BC, new users Y and all users U

1: **procedure** ERPSolver $UA, AM, UCond, UC, SC, BC, Y, U$
2: $CIRCUIT_{formula}$ = ERPtoCircuitSAT(UA, AM, $UCond$, UC, SC, BC, Y, U)
3: CNF_{ERP} = TseytinTransformation($CIRCUIT_{formula}$)
4: Ans = SATSolver(CNF_{ERP})
5: Output Ans
6: **end procedure**

Algorithm 2. Algorithm to Reduce ERP to Circuit SAT

Input: user assignment relation UA, authorization matrix AM, user conditions $UCond$, user capability constraint UC, SoD constraints SC, BoD constraints BC, new users Y and all users U

1: **procedure** ERPtoCircuitSAT (UA, AM, $UCond$, UC, SC, BC, Y, U)
2: $enum_UA \leftarrow$ enumerations of the assignments in UA
3: D_1 = BasicConditiontoCNF($enum_{UA}$, Y)
4: D_2 = BoDtoCNF(BoD, AM, $enum_{UA}$, Y)
5: D_3 = SoDtoCNF(BoD, AM, $enum_{UA}$, UC, Y)
6: $D = D_1 + \text{“}\wedge\text{”} + D_2 + \text{“}\wedge\text{”} + D_3$ ▷ the circuit is represented by a boolean formula
7: Output D
8: **end procedure**

Algorithm 3. Procedure to Convert Basic Conditions of ERP to CNF Formula

Input: enumeration of the assignments in the UA matrix $enum_{UA}$ and the set of new users Y

1: **procedure** BasicConditiontoCNF $enum_{UA}$, Y
2: **for** $j \leftarrow enum_{UA}$ **do**
3: **for** $i \leftarrow 1, |Y| - 1$ **do**
4: $D = D + \text{“}(x\text{”} + str(i) + str(j) + \text{“}\oplus x\text{”} + str(i + 1) + str(j) + \text{“}) \vee \text{”}$
5: ▷ D is a variable storing the string of CNF formula
6: ▷ $str()$ is the function to convert numbers to string
7: **end for**
8: $D = D + \text{“}\wedge\text{”}$
9: **end for**
10: Output D
11: **end procedure**

V. SOLVING ERP

A solution to the ERP problem can be obtained by modeling it using the CNF Satisfiability problem (CNF-SAT). CNF-SAT is referred to as the version of the Satisfiability problem where the boolean formula is specified in Conjunctive Normal Form (CNF). A conjunction of clauses is called CNF, where each clause is a disjunction of literals. The procedure *ERPSolver* in Algorithm 1 presents the steps to obtain a solution for the current problem. In Line 2, the procedure *ERPtoCircuitSAT* is called to convert an instance of ERP to an instance of the Circuit SAT problem. The output is then passed as an argument to the procedure *TseytinTransformation* to convert it to an instance of CNF SAT at Line 3. Finally, the CNF SAT instance of the corresponding ERP problem is fed as an input to a SAT solver at Line 4 to obtain a solution. The output obtained is either *True* or *False*. An output with *True* value denotes that the procedure *ERPSolver* has been able to find a solution and the set X of employees can be replaced by the set Y of new employees. Whereas, a *False* output denotes that the set Y of employees cannot cover all the permissions of the set X of

TABLE 5. Variable sets for Example 1.

	av_{11}	av_{12}	av_{21}	av_{22}	av_{31}	av_{32}
u_1	vs_1	×	×	vs_2	vs_3	×
u_2	×	×	×	×	vs_4	vs_5
u_3	×	vs_6	vs_7	×	×	×
u_4	×	vs_8	×	vs_9	×	vs_{10}
u_5	vs_{11}	×	vs_{12}	×	×	vs_{13}

employees and thus, a valid set of user to attribute-value pair assignments for the set of new employees cannot be obtained.

Algorithm 4. Procedure to Convert BoD Constraints of ERP to CNF Formula

Input: the collection of BoD constraints *BoD*, the authorization matrix *AM*, enumeration of the assignments in the *UA* matrix $enum_{UA}$ and the set of new users Y

```
1: procedure BoDToCNF (BoD, AM, enum_UA, Y)
2:   for bc_x ← BoD do                          ▷ for all BoD constraints
3:   D = D + "("
4:   for p_m ← bc_x do
5:                          ▷ for all permissions in a BoD constraint
6:     for uc_l ← AM(p_m) do
7:                              ▷ for all user conditions for p_m in AM
8:       D = D + "("
9:       for av_k ← uc_l do
10:        ▷ for all attribute-value pairs in a user condition
11:        D = D + "("
12:        for j ← enum_{av_k} do
13:                ▷ for all enum corresponding to av_k in UA
14:          D = D + "("
15:          for i ← 1, |Y| do                  ▷ for all new users
16:            D = D + "x" + i + j + " ∨ "
17:          end for
18:        end for
19:        D = D + ") ∧ "
20:      end for
21:      D = D + ") ∨ "
22:    end for
23:    D = D + ") ⊕ "
24:  end for
25:  D = D + ") ∧ "
26:  end for
27:  Output D
28: end procedure
```

Because there exists a polynomial time reduction from a Circuit SAT instance to a CNF SAT instance using Tseytin transformation [8], in this section we present a polynomial time reduction of ERP to Circuit SAT. The basic premise for the reduction is that the assignments in the part of the UA matrix comprising the users to be replaced have to be covered using the set of new users depending upon their attribute capability. For instance, the assignment (u_1, av_{22}) in *UA* of Example 1 can be covered by un_2 or un_3 and not by un_1 as it is not capable of handling the attribute value pair av_{22}. Each variable of the CNF-SAT problem has to represent such conditions which can be answered in true or false. Thus, each

TABLE 6. Set of variables for each assignment in Example 1.

vs_1	$\{x_{11}, x_{21}\}$
vs_2	$\{x_{22}, x_{32}\}$
vs_3	$\{x_{13}, x_{33}\}$
vs_4	$\{x_{14}, x_{34}\}$
vs_5	$\{x_{15}, x_{25}, x_{35}\}$
vs_6	$\{x_{16}, x_{26}, x_{36}\}$
vs_7	$\{x_{27}, x_{37}\}$
vs_8	$\{x_{18}, x_{28}, x_{38}\}$
vs_9	$\{x_{29}, x_{39}\}$
vs_{10}	$\{x_{1\,10}, x_{2\,10}, x_{3\,10}\}$
vs_{11}	$\{x_{1\,11}, x_{2\,11}\}$
vs_{12}	$\{x_{2\,12}, x_{3\,12}\}$
vs_{13}	$\{x_{1\,13}, x_{2\,13}, x_{3\,13}\}$

assignment will be represented by a set of variables depending on the capabilities of the users in the set Y. For example, the set $vs_2 = \{x_{22}, x_{32}\}$ of variables will correspond to the assignment (u_1, av_{22}) as shown in Tables 5 and 6. Here, for each x_{ij} and vs_j, j represents the variable corresponding to the jth assignment in *UA* and i represents the ability of the ith user in the set Y covering it. Tables 5 and 6 describe all the sets of variables for all the given assignments in Example 1. As presented in the procedure *ERPtoCircuitSAT* of Algorithm 2, the rest of the reduction has to be carried out for the following three parts.

- Basic condition of the problem (function call at Line 3)
- BoD constraints (function call at Line 4)
- SoD constraints (function call at Line 5)

These steps will generate digital circuit segments D_1, D_2 and D_3, respectively. Finally, the segments are joined as $D_1 \wedge D_2 \wedge D_3$ using AND gates in Line 6.

Basic Condition of the Problem. For each assignment in *UA*, a set of variables is considered depending on the capabilities of the new users to handle the particular assignment. However, only a single user from the set Y is required to cover a particular assignment. Therefore, $\forall j \in enum_{UA}$, only one variable in the set $\{x_{1j}, ..., x_{|Y|j}\}$ is required to be assigned to 1, here $enum_{UA}$ is the set of enumerations of the assignments in the matrix *UA*. The Boolean formula enforcing this condition is :

$$D_1 = \bigwedge_{\forall j \in enum_{UA}} \big((x_{1j} \oplus x_{2j}) \vee \ldots \vee (x_{1j} \oplus x_{|Y|j}) \vee \ldots \vee (x_{(|Y|-1)j} \oplus x_{|Y|j})\big). \quad (1)$$

For instance, the segment of the equation corresponding to the assignments in the second row (for u_2) of the *UA* matrix in Example 1 is $(x_{14} \oplus x_{34}) \wedge \big((x_{15} \oplus x_{25}) \vee (x_{15} \oplus x_{35}) \vee (x_{25} \oplus x_{35})\big)$. The constructed Boolean formula can be easily transformed into a digital circuit. The steps to encode the basic conditions of the problem into a boolean formula is presented in the procedure *BasicConditiontoCNF* of Algorithm 3. The *for* loop in Line 2 to Line 9 takes care

of the iteration for each assignment in the UA relation. Line 3 iterates over the number of new users in set Y. With the help of these iterations, Line 4 constructs each clause shown in Equation 1, which are finally joined using the AND operator in Line 8 to construct the boolean formula D_1.

Algorithm 5. Procedure to Convert SoD Constraints of ERP to CNF Formula

Input: the collection of SoD constraints SoD, the authorization matrix AM, enumeration of the assignments in the UA matrix $enum_{UA}$, user capability constraint UC and the set of new users Y

1: **procedure** SoDtoCNF (BoD, AM, $enum_{UA}$, UC, Y)
2: **for** $PR \leftarrow SoD$ **do** ▷ for all SoD constraints
3: $D = D +$ "("
4: **for** $i \leftarrow 1, |Y|$ **do** ▷ for all new users
5: **for** $j \leftarrow 1, |PR|$ **do**
6: ▷ for all permissions in the SoD constraint
7: $D = D +$ "("
8: **for** $k \leftarrow AM(p_j)$ **do**
9: ▷ for all user conditions for p_j in AM
10: $D = D +$ "("
11: **for** $l \leftarrow uc_k$ **do** ▷ for all av pairs in uc_k
12: **if** y_i is capable for av_l in UC **then**
13: ▷ checking for UC constraint
14: **for** $m \leftarrow enum_{av_l}$ **do**
15: ▷ for all enums corresponding to av_l
16: $D = D +$ "x" $+ str(i) + str(x) +$ " $\vee$ "
17: **end for**
18: **end if**
19: $D = D +$ "$) \wedge$ "
20: **end for**
21: $D = D +$ "$) \vee$ "
22: **end for**
23: $D = D +$ "$)CSA$"
24: ▷ Carry Save Adder as shown in Figure 2
25: **end for**
26: **end for**
27: $D = D +$ "$)COMP$" $+ (k - k')$
28: ▷ Comparator as shown in Figure 4
29: ▷ calculation of k' is shown in Figure 3
30: $D = D +$ " $\wedge$ " ▷ Anding all SoDs
31: **end for**
32: Output D
33: **end procedure**

BoD Constraints. The Boolean formula enforcing each BoD constraint $bc_x = \{p_1, ..., p_o\}$ is:

$$D_2 = \oplus_{\forall p_o \in bc_x} \left(\bigvee_{\forall uc_l \in AM(p_o)} \left(\bigwedge_{\forall av_k \in uc_l} \left(\bigvee_{\substack{j \in enum_{av_k} \\ i \in \{1,...,|Y|\}}} x_{ij} \right) \right) \right). \tag{2}$$

Here, $AM(p_o)$ is the set of user conditions corresponding to the permission p_o in the authorization matrix AM and $enum_{av_k}$ is the set of enumerations of the assignments corresponding to av_k in the matrix UA.

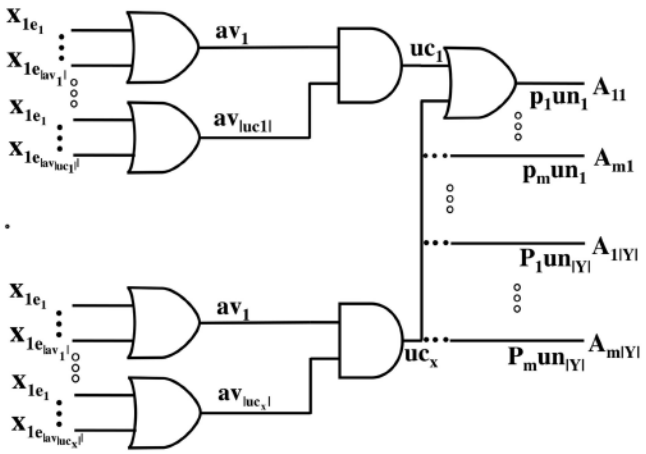

FIGURE 1. Circuit segment for permission and new user pair of each SoD constraint.

For instance, the segment of the equation for p_4 of bc_2 in Example 1 is $(x_{27} \vee x_{37} \vee x_{2\,12} \vee x_{3\,12}) \wedge (x_{15} \vee x_{25} \vee x_{35} \vee x_{1\,10} \vee x_{2\,10} \vee x_{3\,10} \vee x_{1\,13} \vee x_{2\,13} \vee x_{3\,13})$. Such equation segments are constructed for each permission in the BoD constraint and are joined together with a $\oplus$ operator. For each BoD constraint, the constructed Boolean formula is converted into a digital circuit, and these circuit segments are joined using AND gates to form the whole segment for the set of BoD constraints.

The procedure *BoDtoCNF* in Algorithm 4 presents this part of the reduction. While Line 2 iterates the procedure over each BoD constraint, Line 4 enforces the iteration over all the permissions for a particular BoD constraint. All the user conditions in AM for the particular permission currently in process under the loop is considered in Line 6. For a particular user condition, all the attribute-value pairs in it is considered in Line 9. Line 12 iterates the rest of the code over each assignment in UA for a particular attribute-value pair, and the iteration for all the new users is considered in Line 15. Using the described iterations, the Boolean formula D_2 is constructed in Lines 3, 8, 11, 14, 16, 19, 21, 23 and 25. Finally, the output of the algorithm is returned in Line 27.

SoD constraints: For each SoD constraint $\langle \{p_1, ..., p_m\}, k \rangle$, the corresponding digital circuit segment D_3 is shown in Figures 1–5. Since an SoD constraint is dependent on the set Y of new users as well as on the set $U - X$ of old users, the circuit corresponding to an SoD constraint is divided into two parts: the first part is illustrated in Figures 1 and 2, whereas the second part is presented in Figure 3. In Figure 1, $\{e_1, ..., e_{|av_i|}\}$ are the elements of the set $enum_{av_i}$. To consider the dependency of the users not to be replaced on the SoD constraints, the circuit in Figure 3 calculates the number of users in the set $U - X$, which are dependent on the corresponding SoD constraints. Here, y_{ij} denotes a boolean value which is *True* if the ith user of the set $U - X$ can get the jth permission in the set $\{p_1, ..., p_m\}$. It is to be noted that the value of y_{ij} can easily be obtained from the given UA matrix, AM matrix and user conditions. The output B corresponding to the new users is compared with the integer $k - k'$ in the circuit segment shown in Figure 4. Finally, the segments for each SoD constraint are joined using an AND gate in

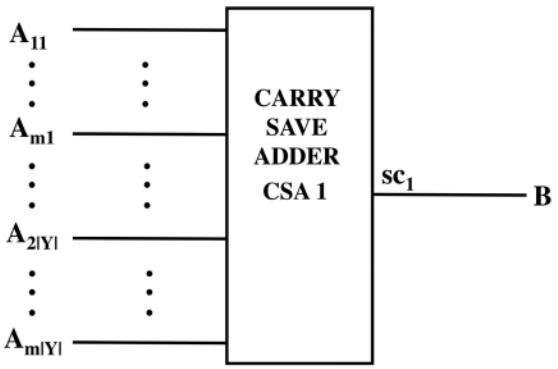

FIGURE 2. Outputs of the circuit in Figure 1 are added using carry save adder.

Figure 5 to form the circuit for the whole set of SoD constraints. The procedure *SoDtoCNF* in Algorithm 5 presents this part of the reduction. While Line 2 iterates the rest of the algorithm over all the SoD constraints, iteration for all the new users is taken care of in Line 4. Line 5 initiates a for loop for all the permissions in the current SoD constraint, and Line 8 considers all the user conditions in *AM* corresponding to the permission considered in the preceding for loop. The set of attribute value pairs is considered in Line 11 for the current user condition. Line 12 enforces the *UC* constraint. Line 14 loops the rest of the code for each assignment in *UA* for the particular attribute-value pair considered in the preceding for loop. The rest of the lines up to Line 22 construct the Boolean output $\{A_{11}, ..., A_{m1}, ..., A_{1|Y|}, ..., A_{m|Y|}\}$ as shown in the circuit diagram of Figure 1. For instance, A_{11} corresponding to p_1 of sc_3 in Example 1 is $\big((x_{16} \vee x_{26} \vee x_{36} \vee x_{18} \vee x_{28} \vee x_{38}) \wedge (x_{27} \vee x_{37} \vee x_{2\,12} \vee x_{3\,12})\big) \vee \big((x_{16} \vee x_{26} \vee x_{36} \vee x_{18} \vee x_{28} \vee x_{38}) \wedge (x_{15} \vee x_{25} \vee x_{35} \vee x_{1\,10} \vee x_{2\,10} \vee x_{3\,10} \vee x_{3\,13} \vee x_{2\,13} \vee x_{3\,13})\big)$. The output is then fed to a Carry Save Adder in Line 23 as shown in Figure 2. Line 27 implements the functioning of the circuit diagram in Figures 3 and 4. Finally, the boolean formula of the corresponding circuit for each SoD constraint is joined using an AND gate as shown in Figure 5 in Line 30.

Time complexity:

The worst case time complexities of the individual procedures in Algorithms 3, 4 and 5 are as follows.

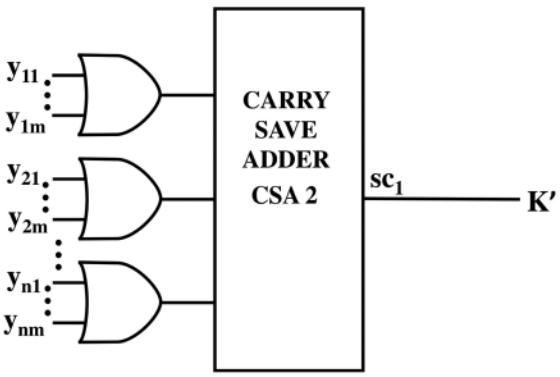

FIGURE 3. Circuit segment corresponding to the dependence of an SoD constraint on old users.

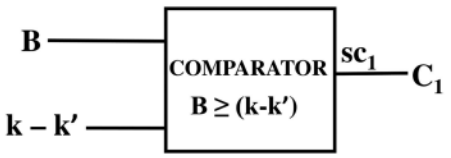

FIGURE 4. Outputs of circuits in Figures 2 and 3 are added using CSA adder and compared with the integer k.

- *BasicConditiontoCNF*: $\mathcal{O}(|enum_{UA}| \times |Y|)$
- *BoDtoCNF*: $\mathcal{O}(|BC| \times |Y| \times |AM| \times |UA|)$, here $|AM|$ and $|UA|$ represents the total number of cells in the *AM* and *UA* matrix respectively
- *SoDtoCNF*: $\mathcal{O}(|SC| \times |Y| \times |AM| \times |UA|)$

The overall running time complexity of the procedure *ERPtoFormula* Algorithm 2 turns out to be $\mathcal{O}(|Y| \times |UA| \times |AM| \times Cons)$, where *Cons* represents the total number of SoD and BoD constraints. It is to be noted that the worst case time complexity of calculating k' for each SoD constraint using the digital circuit shown in Figure 3 is $\mathcal{O}(|U| \times |Per|)$. We have assumed that the k' values for each SoDs are available prior to the execution of *SoDtoCNF*.

It is to be noted that the procedure *ERPSolver* in Algorithm 1 can also be used to solve the employee replacement problem in the context of non-ABAC access control models. This flexibility of the solution is mainly due to the fact that DAC, MAC and RBAC models are special cases of ABAC. To solve ERP instances for these three access control models using the algorithms presented, simple reductions as described below have to be carried out. For an ERP instance with respect to a DAC system, the sets X and Y of users will remain unchanged, the set AV of the attribute value pairs will be the set of permissions in the system, i.e., $AV = Per$. For each attribute value pair, a user condition is defined for the instance. Therefore, the *AM* and the *UA* matrices will turn out to be diagonal matrices, and the *UC* matrix will effectively turn out to be a user to permission assignment relation denoting capabilities of the set Y of users.

For a MAC system also, the sets X and Y of users will remain unchanged. However, the set AV will be the set of possible security clearances and for each attribute value pair, a user condition has to be considered. For appropriate configuration of the set of permissions in the instance, the object attributes will be the set of possible security labels of the resources and for each object attribute-value pair, an object condition has to be considered. Thus, the *AM* matrix would effectively be a clearance to permission assignment relation and the properties of the MAC model particular to the Bell LaPadula or the Biba model [5] can be specified explicitly

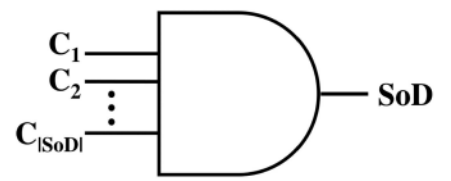

FIGURE 5. Circuit segments of each SoD constraint are joined using an AND gate.

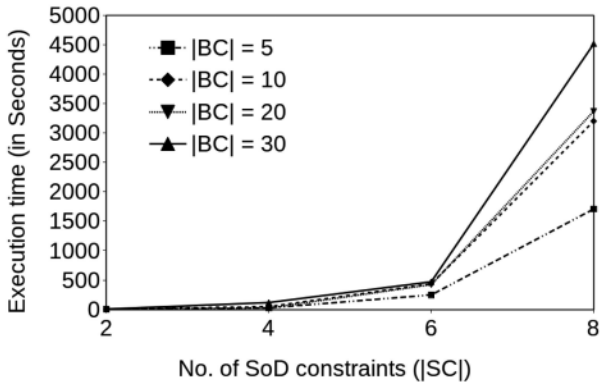

FIGURE 6. Variation of execution time with respect to $|SC|$ for different values of $|BC|$.

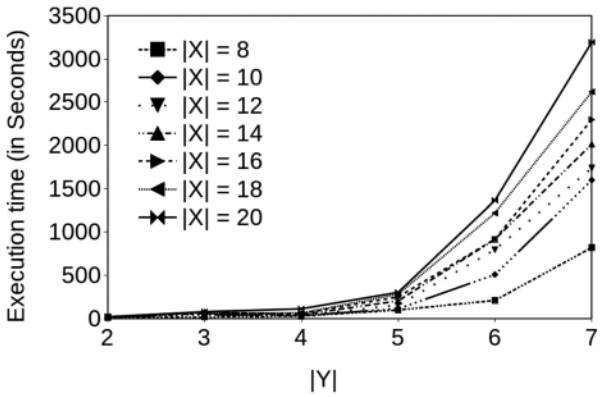

FIGURE 7. Variation of execution time with respect to $|Y|$ for different values of $|X|$.

using this relation. The *UA* matrix would work as a user to clearance assignment relation where the clearances of the subjects can be specified. The *UC* matrix would effectively be a user to clearance assignment relation for the set Y of users which can be used to specify the capability clearances that the new users can acquire in the system.

For an RBAC system as well, the sets X and Y of users will remain unchanged. However, the set AV will be the set of roles, and for each attribute value pair, a user condition is defined. Thus, the *AM* matrix effectively will be a role to permission relation and the *UA* matrix will be a user to role relation of the considered RBAC system. The user capability constraint *UC* for the set Y of users will turn out to be a user to role assignment relation in this case.

VI. EXPERIMENAL EVALUATION

The modeling described in Section V was implemented using Python, the SAT solver used for solving the resultant CNF-SAT instance is CryptoMiniSAT.[1] The experimentation to test the implementation was carried out on an Intel Xeon E5-2697 v2 (processor speed of 2.7 GHz) server with 256 GB of RAM running Linux. The execution time required for varying number of BoD constraints, number of SoD constraints, $|X|$, $|Y|$, and number of attribute-value pairs for an ABAC system with 1000 users is studied. Synthetic datasets were generated randomly and twenty different runs were carried out for each combination of parameters. The execution time reported in this section is the mean over twenty runs. It is to be noted that for the sake of simplicity in implementation, the Carry Save Adder, Ripple Carry Adder and the Comparator blocks in the boolean circuit of the corresponding Circuit-SAT instance of the given ERP were converted to circuit blocks of only AND, OR, NOT and XOR before passing them for the Tseytin transformation.

Figure 6 presents the variation in execution time with varying number of SoD constraints from 2 to 8 and with varying number of BoD constraints from 5 to 30, keeping $|X|$ as 10, $|Y|$ as 5, number of attribute-value pairs as 12, number of user conditions as 3 and number of permissions as 10. We observe that the execution time increases cubically as the number of SoD constraints increases. The corresponding circuit size of the Circuit-SAT instance increases as the repetitions of the blocks shown in Figures 1, 2, 3, 4 increase with increase in the number of SoD constraints. This, in turn, increases the number of variables and clauses in the CNF-SAT instance causing the observed trend in the execution time. For varying number of BoD constraints, it is observed that there is a linear increase in execution time. The increase is because of the increase in size of the digital circuit which in turn increases the number of variables and clauses in the resultant CNF formula provided as an input to the SAT solver.

Figure 7 shows the variation in execution time while varying $|Y|$ from 2 to 7 and $|X|$ from 8 to 20, keeping the number of attribute-value pairs as 25, number of user conditions as 4, number of permissions as 5 and number of SoD and BoD constraints as 5. The execution time is observed to cubically increase with increase in $|Y|$. This trend is because of the multiple increases in the number of variables with increase in the size of new users. It is also observed that the execution time increases when $|X|$ increases from 10 to 20. This is again because the number of variables increases as the number of assignments in the *UA* matrix increases with increase in the size of the set X.

Figure 8 presents the variation in execution time with varying number of attribute-value pairs from 15 to 40, keeping $|X|$ as 10, $|Y|$ as 5, number of user conditions as 4, number of permissions as 5, number of SoD constraints as 5 and number of BoD constraints as 5. It is observed that the execution time increases with increase in the number of attribute-value pairs. This is because the number of assignments in the *UA* matrix increases with an increase in the number of attribute-value pairs, causing the number of variables to increase.

Figure 9 presents the variation in execution time with varying number of permissions, keeping $|X|$ as 20, $|Y|$ as 3, number of attribute-value pairs as 25, number of user conditions as 4, number of SoD constraints as 5 and number of BoD constraints as 5. It is observed that the execution time increases with increase in the number of permissions. It is to be noted that with an increase in the number of permissions, the size of the

[1]https://www.msoos.org/cryptominisat2

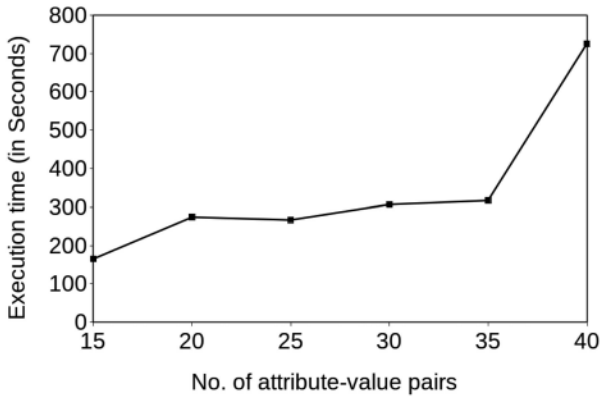

FIGURE 8. Variation of execution time with respect to number of attribute-value pairs.

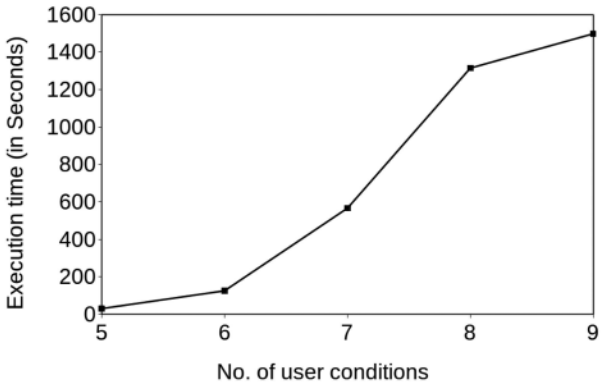

FIGURE 10. Variation of execution time with respect to number of user conditions.

set *PR* tends to rise which causes the size of the circuit block in Figure 1 to increase. As a result, the number of clauses and the size of the clauses increase in the resulting Boolean formula, increasing the overall execution time.

Figure 10 presents the variation in execution time with varying number of permissions, keeping $|X|$ as 20, $|Y|$ as 3, number of attribute-value pairs as 25, number of permissions as 5, number of SoD constraints as 5 and number of BoD constraints as 5. It is observed that the execution time increases with increase in the number of user conditions. This is because of the increase in the size of the Boolean formula constructed corresponding to the BoD constraints.

To directly evaluate the *ERPtoCircuitSAT* procedure proposed in this work, the variation of execution time only for this step (Line 2) of Algorithm 1 is presented in Figure 11. The dataset used is the same as the one used for the experiment presented in Figure 7. We observe a similar trend in the variation. However, on comparing Figure 11 with Figure 7, it is observed that the time required to execute the procedure *ERPtoCircuitSAT* is quite small in comparison to the total required execution time for solving an ERP instance. Thus, the main complexity of execution comes from the well established procedures *TseytinTransformation* and *SATSolver* that is used in the solution. This also gives an indication that the overall performance of Algorithm 1 with respect to execution time can be improved if more efficient SAT solvers and algorithms for transforming a digital circuit to a CNF boolean formula can be developed.

It is to be noted that in real world business organizations, the plan for instating replacements for retired or retrenched employees is conducted immediately after the vacancies are created so as to minimally affect the smooth functioning of the organization. Delay in filling in of vacancies may adversely effect the business productivity, revenue, existing employee engagement and company reputation [28]. The number of employees leaving the organization at a time or within a span of short duration is generally not more than what is considered in the experiment presented in Figure 7. Moreover, even if the number of employees leaving the organization is large, they can be divided into batches of smaller groups of employees before carrying out the verification for its replacement with a smaller set of new employees. Hence, the implementation environment presented in this section is fairly representative and practical.

VII. RELATED WORK

There is a fairly large body of work in the literature in the area of workforce optimization and workforce downsizing. However, none of these deal with the problem of deciding the viability of replacing the set of downsized or retired employees with a smaller set of employees in a constrained

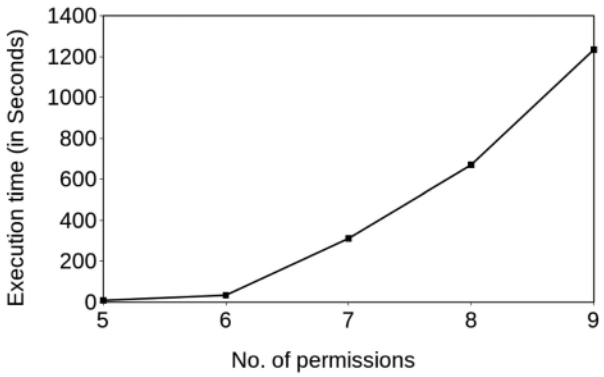

FIGURE 9. Variation of execution time with respect to number of permissions.

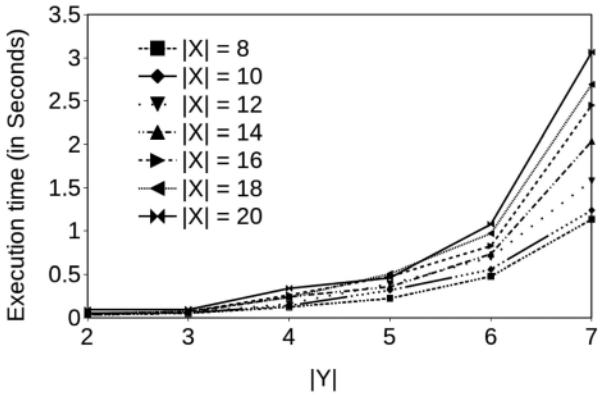

FIGURE 11. Variation of execution time for the *ERPtoCircuitSAT* procedure with respect to $|X|$ and $|Y|$.

ABAC system. On the other hand, there is some work in the area of access control under situations having certain similarities with ours. In this section, we discuss such related work and point out their key differences.

An assortment of access control models have been studied and implemented for enforcing security in commercial information systems [5], [35]. Although RBAC is still favored by a fairly large number of organizations [13], [44], [45], and efforts have been made to increase its deployability [16], [24], [41], its lack of extendibility and other limitations have led to the recent proliferation of ABAC [19]. DAC, MAC and RBAC have been shown to be special cases of ABAC and thus, considering ABAC also provides a solution for the other access control models. A range of access control models based on ABAC have been introduced in the last few years. A framework to develop a distributed access control architecture, combining role and attribute was proposed by Qi *et al.* [27]. Jin *et al.* propose a formal definition of a generalized ABAC model that has the capability to configure DAC, MAC and RBAC, thus, formally establishing a connection between ABAC and the other three classical access control models [21].

A location-aware ABAC model that can help in detecting security infringements in online social networks has been presented in [18]. Servos *et al.* formally defined a hierarchical model of ABAC which includes the property of attribute inheritance [37]. A scheme which uses access tree made up of logic expressions over attribute to define the access control structures has been proposed by Chatterjee *et al.* [7]. Biswas *et al.* propose a policy enumeration model named LaBAC (Label-Based Access Control) in [6]. This model uses an object attribute and a user attribute for enumeration of authorization policies. The ABAC model for cloud environment discussed by Riad *et al.* in [29] supports attribute-rules for access control. A comprehensive definition of the ABAC model has been published by NIST [19]. Further research carried out to effectively deploy ABAC in different types of commercial information systems can be found in [3], [11], [17], [38], [43].

Various constraints necessary to enforce different security policies in information systems are supported by access control models. As mentioned before, SoD and BoD are two of the most powerful and widely used constraints. SoD, which has long been present in physical systems in the name of "two-man rule", was introduced by Clark and Wilson as a procedure for controlling fraud and errors in information systems [5]. It was observed by the authors that separating all operations into a set of tasks and compelling each task to be executed by a different subject can enforce external consistency. Taking correspondence from the traditional definition of SoD, in this work, a general definition of SoD is considered which necessitates the involvement of at least a particular number of subjects to complete a job that requires a specified set of permissions.

The BoD constraints enforce a subject to perform none of or all of the task in a specified set, thus defining a relation between them [39]. In other words, to complete a job, the set of "bound tasks" is compelled to be performed by the same subject. The BoD constraints that we have considered in this work enforce a subject to acquire all the permissions in a set in case she requires any one of them.

Another type of constraint that is used in the current work is user capability constraint, which is said to be satisfied if a user is only assigned to attributes which she is capable of acquiring. User capability constraint was used in our previous work [33]. However, there the capabilities of the users were defined in terms of the roles in an RBAC system. Although the term "capability" is not new in access control systems, it has been used in entirely different contexts. Gusmeroli *et al.* [15] proposed the capability-based access control system for a distributed environment, where the subject has to present her assigned permissions (termed as "authorization capabilities") to the service provider to get access to the requested resource to perform the requested operation. Similarly, in the capability based computer system [5], access rights assigned to a user are referred to as her capability. The term capability in the current context is used to specify the ability of an employee to handle different tasks based on her skills acquired through training or by experience prior to the assignment.

We considered the question of workforce optimization under access control systems in some of our previous work [30], [31], [32], [33]. In [30] and [31], the problem of determining the minimum number of users necessary to enforce Statically Mutually Exclusive Roles constraints in RBAC was identified and solutions were proposed. In presence of cardinality and mutually exclusive task constraints, the problem of computing the least number of users required to perform a given set of jobs was considered in [32]. This work, however, did not assume any particular access control model to be in place. In [33], we studied the problem of maximizing the utilization of workforce in an RBAC system with relevant security constraints. Here, maximization of the utilization was achieved by maximizing the number of assignments for employees to roles under the specified security constraints. It is to be noted that, none of our previous work considered the problem of optimizing the workforce while downsizing. Moreover, none of the previous work considered an ABAC system.

There has been some prior work on workflow management dealing with the assignment of users to her authorized rights over resources [4], [10], [42]. For a given workflow, Crampton [9] first analyzed the computational complexity of deciding whether an authorization policy is satisfied by a user to task allocation assignment. All such work deal with the problem of workflow satisfiability, that is, verifying whether there exists a user to authorization right assignment relation satisfying all the given security constraints. It is to be noted that, they handle an entirely different aspect, because they do not consider the problem of realization of the existing assignment relation with lesser number of users. Moreover, the fact that the problems dealt with were not compatible with the ABAC model and the presence of hierarchy within the tasks make those entirely different from the current problem.

The problem that looks most similar to our work is the one discussed by Basin *et al.* [2]. The authors considered the problem of optimizing the transition of an existing assignment relation of users to tasks that do not necessarily satisfy all the specified security constraints to a new assignment that allows a successful workflow execution while satisfying all of them. The optimization factor considered in this problem is the cost of transition which includes administrative cost, maintenance cost and risk. The fundamental differences of ERP with this problem is as follows. First, ERP is a decision problem, whereas Basin *et al.* consider an optimization problem. Even if we consider an optimization version of ERP, the notion of optimality would differ substantially, as it would optimize the number of employees rather than the cost of transition. The number of users in the new user-task assignment relation is the same as the existing one, whereas we consider the new assignment with a lesser number of users. Even using a lesser number of users, ERP requires to restore all the assignments in the existing user to attribute assignment relation. In the problem considered by Basin *et al.*, the old assignments may not get restored and new assignments might get introduced. While we consider arbitrarily large k-n SoD constraints, Basin *et al.* consider a form of SoD constraint which is basically equivalent to a set of 2-2 SoD constraints. Finally, the problem considered in our work is not dependent on any particular workflow execution and considers the ABAC model.

There is some work in the field of management research that deal with the issue of downsizing in an organization [12], [34], [36]. It has been discussed in these papers that there are broadly three strategies that different firms adopt to downsize: workforce reduction, work redesign, and systemic strategy. Considering the different downsizing strategies, the workforce reduction approach is the first choice for most of the organizations [36]. Among the various tactics of workforce reduction, succession planning has been considered a way which is quite close to the approach considered in this work. However, no algorithmic or mathematical tool has been developed to facilitate downsizing in any previous work. Thus, the problem addressed by us is unique and the approach innovative, providing a solution to an important information management challenge.

VIII. CONCLUSION

In this article, we consider the situation where employees may turn out to be non-productive and have to be replaced by a fresh set of employees, while maintaining security constraints. A similar condition also occurs when a set of retiring employees has to be replaced by a smaller set of new employees in order to optimize the workforce. The problem has been formally defined as the Employee Replacement Problem, and shown to be NP-hard. To obtain a solution, the problem was modeled using the CNF-SAT problem. An instance of Circuit-SAT was considered as an intermediate step and Tseytin transformation was used subsequently to obtain a CNF-SAT instance. We have implemented the proposed approach and experimentally validated it over varying datasets of different sizes. It is observed that the proposed solution is reasonably efficient and practical for real world business information systems.

In the future, we plan to expand our study in several directions. It would be interesting to explore the possibility where an organization using ABAC retrenches a set of nonproductive employees and suggests some existing employees to acquire necessary skills to carry out the responsibilities of the retrenched employees. We would also like to study the optimization version of ERP that deals with finding the smallest set of employees to replace a particular set of nonproductive employees.

ACKNOWLEDGMENTS

Research reported in this publication was supported by the National Institutes of Health under award R01GM118574 and by the National Science Foundation under awards CNS-1564034, CNS-1624503, and CNS-1747728. The content is solely the responsibility of the authors and does not necessarily represent the official views of the agencies funding the research.

REFERENCES

[1] B. Goren, "Five steps to optimizing human capital," *SAS Inst. Inc.*, 2008. [Online]. Available: www.sas.com/resources/asset/5stepstooptimization.pdf
[2] D. Basin, S. J. Burri, and G. Karjoth, "Optimal workflow-aware authorizations," in *Proc. ACM Symp. Access Control Models Technol.*, 2012, pp. 93–102.
[3] Y. Benkaouz, M. Erradi, and B. Freisleben, "Work in progress: K-nearest neighbors techniques for abac policies clustering," in *Proc. ACM Int. Workshop Attribute Based Access Control*, 2016, pp. 72–75.
[4] E. Bertino, E. Ferrari, and V. Atluri, "The specification and enforcement of authorization constraints in workflow management systems," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 1, pp. 65–104, 1999.
[5] M. Bishop, *Computer Security*. Reading, MA, USA: Addison-Wesley, 2018.
[6] P. Biswas, R. Sandhu, and R. Krishnan, "Label-Based Access Control: An abac model with enumerated authorization policy," in *Proc. ACM Workshop Attribute Based Access Control*, 2016, pp. 1–12.
[7] S. Chatterjee, A. K. Gupta, V. K. Mahor, and T. Sarmah, "An efficient fine grained access control scheme based on attributes for enterprise class applications," in *Proc. Int. Conf. Signal Propag. Comput. Technol.*, 2014, pp. 273–278.
[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.
[9] J. Crampton, "A reference monitor for workflow systems with constrained task execution," in *Proc. 10th ACM Symp. Access Control Models Technol.*, 2005, pp. 38–47.
[10] J. Crampton, G. Gutin, and A. Yeo, "On the parameterized complexity and kernelization of the workflow satisfiability problem," *ACM Trans. Inf. Syst. Secur.*, vol. 16, no. 1, pp. 4:1–4:31, 2013.
[11] D. Ferraiolo, R. Chandramouli, R. Kuhn, and V. Hu, "Extensible access control markup language (xacml) and next generation access control (ngac)," in *Proc. ACM Int. Workshop Attribute Based Access Control*, 2016, pp. 13–24.
[12] G. Franco, "Workforce Downsizing: Strategies, archetypes, approaches and tactics," *J. Manage. Res.*, vol. 13, no. 2, pp. 67–76, 2013.
[13] L. Fuchs, G. Pernul, and R. Sandhu, "Roles in information security–a survey and classification of the research area," *Comput. Secur.*, vol. 30, no. 8, pp. 748–769, 2011.
[14] Gartner IAM 2020 Predictions, "Attributes Are Now "How We Role"," 2011. [Online]. Available: https://www.avatier.com/products/identity-management/resources/gartne r-iam-2020-predictions/
[15] S. Gusmerolia, S. Piccionea, and D. Rotondi, "A capability-based security approach to manage access control in the internet of things," *Math. Comput. Model.*, vol. 58, no. 5–6, pp. 1189–1205, 2013.

[16] P. Harika, M. Nagajyothi, J. C. John, S. Sural, J. Vaidya, and V. Atluri, "Meeting cardinality constraints in role mining," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 1, pp. 71–84, 2014.

[17] M. Hffmeyer and U. Schreier, "Restacl - An access control language for restful services," in *Proc. ACM Int. Workshop Attribute Based Access Control*, 2016, pp. 58–67.

[18] A. C. Hsu and I. Ray, "Specification and enforcement of location-aware attribute-based access control for online social networks," in *Proc. ACM Int. Workshop Attribute Based Access Control*, 2016, pp. 25–34.

[19] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, "Guide to attribute based access control (ABAC) definition and considerations," *NIST Special Publication*, pp. 1–37, 2014. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-162.pdf

[20] S. Jha, S. Sural, V. Atluri, and J. Vaidya, "Specification and verification of separation of duty constraints in attribute-based access control," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 4, pp. 897–911, Apr. 2018.

[21] X. Jin, R. Sandhu, and R. Krishnan, "A unified attribute-based access control model covering DAC, MAC and RBAC," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*, 2012, pp. 41–55.

[22] J. Li, N. Chen, and Y. Zhang, "Extended file hierarchy access control scheme with attribute based encryption in cloud computing," *IEEE Trans. Emerg. Topics Comput.*, Mar. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8666072, doi: 10.1109/TETC.2019.2904637.

[23] N. Li, M. V. Tripunitara, and Z. Bizri, "On mutually exclusive roles and separation-of-duty," *ACM Trans. Inf. Syst. Secur.*, vol. 10, no. 2, pp. 1–36, 2007.

[24] H. Lu, J. Vaidya, V. Atluri, and Y. Hong, "Constraint-aware role mining via extended boolean matrix decomposition," *IEEE Trans. Dependable Secure Comput.*, vol. 9, no. 5, pp. 655–669, Sep./Oct. 2012.

[25] D. V. Miller and R. Baldwin, "Access control by boolean expression evaluation," in *Proc. Annu. Comput. Secur. Appl. Conf.*, 1990, pp. 131–139.

[26] C. Morisset, T. A. C. Willemse, and N. Zannone, "A framework for the extended evaluation of ABAC policies," *Cybersecurity*, vol. 2, no. 1, 2019, Art. no. 6.

[27] H. Qi, H. Ma, J. Li, and X. Di, "Access control model based on role and attribute and its applications on space-ground integration networks," in *Proc. Int. Conf. Comput. Sci. Netw. Technol.*, 2015, pp. 1118–1122.

[28] R. Half, "What are the Risks of a Delayed Recruitment Process," 2017. [Online]. Available: www.talentlyft.com/en/blog/article/203/what-are-the-risks-of-a-delayed -recruitment-process

[29] K. Riad, Z. Yan, H. Hu, and G.-J. Ahn, "AR-ABAC: A new attribute based access control model supporting attribute-rules for cloud computing," in *Proc. IEEE Conf. Collaboration Internet Comput.*, 2015, pp. 28–35.

[30] A. Roy, S. Sural, and A. K. Majumdar, "Minimum user requirement in role based access control with separation of duty constraints," in *Proc. 12th Int. Conf. Intell. Syst. Des. Appl.*, 2012, pp. 386–391.

[31] A. Roy, S. Sural, and A. K. Majumdar, "Impact of multiple t-t SMER constraints on minimum user requirement in RBAC," in *Proc. Int. Conf. Inf. Syst. Secur.*, 2014, pp. 109–128.

[32] A. Roy, S. Sural, A. K. Majumdar, J. Vaidya, and V. Atluri, "Minimizing organizational user requirement while meeting security constraints," *ACM Trans. Manage. Inf. Syst.*, vol. 6, no. 3, pp. 12:1–12:25, 2015.

[33] A. Roy, S. Sural, A. K. Majumdar, J. Vaidya, and V. Atluri, "On optimal employee assignment in constrained role-based access control systems," *ACM Trans. Manage. Inf. Syst.*, vol. 7, no. 4, pp. 10:1–10:24, 2016.

[34] L. Ryan and K. A. Macky, "Downsizing organizations: Uses, outcomes and strategies," *Asia Pacific J. Human Resour.*, vol. 36, no. 2, pp. 29–45, 1998.

[35] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *IEEE Comput.*, vol. 29, no. 2, pp. 38–47, Feb. 1996.

[36] R. Schmenner and C. Lackey, ""Slash and Burn" doesn't kill weeds: Other ways to downsize the manufacturing organization," *Bus. Horizons*, vol. 37, no. 4, pp. 80–87, 1994.

[37] D. Servos and S. L. Osborn, "HGABAC: Towards a formal model of hierarchical attribute-based access control," in *Proc. Int. Symp. Found. Pract. Secur.*, 2014, pp. 187–204.

[38] D. Servos and S. L. Osborn, "Current research and open problems in attribute-based access control," *ACM Comput. Surveys*, vol. 49, no. 4, pp. 65:1–65:45, 2017.

[39] M. Strembeck and J. Mendling, "Generic algorithms for consistency checking of mutual-exclusion and binding constraints in a business process context," in *Proc. OTM Confederated Int. Conf. "On Move Meaningful Internet Syst."*, 2010, pp. 204–221.

[40] D. Thilakanathan, S. Chen, S. Nepal, and R. Calvo, "Safeprotect: Controlled data sharing with user-defined policies in cloud-based collaborative environment," *IEEE Trans. Emerg. Topics Comput.*, vol. 4, no. 2, pp. 301–315, Apr.-Jun. 2015.

[41] J. Vaidya, V. Atluri, J. Warner, and Q. Guo, "Role engineering via prioritized subset enumeration," *IEEE Trans. Dependable Secure Comput.*, vol. 7, no. 3, pp. 300–314, Jul.-Sep. 2008.

[42] Q. Wang and N. Li, "Satisfiability and resiliency in workflow authorization systems," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 4, pp. 40:1–40:35, 2010.

[43] Z. Xu and S. D. Stoller, "Mining attribute-based access control policies," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 5, pp. 533–545, Oct. 2015.

[44] X. Zhao, C. Liu, S. Yongchareon, M. Kowalkiewicz, and W. Sadiq, "Role-based process view derivation and composition," *ACM Trans. Manage. Inf. Syst.*, vol. 6, no. 2, pp. 7:1–7:24, 2015.

[45] H. Zhu and M. Zhou, "Roles in information systems: A survey," *IEEE Trans. Syst., Man Cybern., Part C: Appl. Rev.*, vol. 38, no. 3, pp. 377–396, May 2008.

ARINDAM ROY received the bachelor's degree in computer science and engineering from the West Bengal University of Technology, West Bengal, India, and the master's degree in information technology from IIT Kharagpur, India. He is working toward the the PhD degree in Advanced Technology Development Center, IIT Kharagpur. He is currently a senior lecturer of big data analytics with Goa Institute of Management. His research interests include access control and workforce optimization.

SHAMIK SURAL received the PhD degree from Jadavpur University, Kolkata, India. He is a professor with the Department of Computer Science and Engineering, IIT Kharagpur, India. His research interests include computer security, data mining and database systems. He has published extensively in reputed international journals and conferences. He is a senior member of the IEEE and has previously served as the Chairman of the IEEE Kharagpur Section.

ARUN KUMAR MAJUMDAR received the undergraduate and master's degrees from the University of Calcutta, and the PhD degree from the University of Florida, Gainesville. He retired from the position as a professor of the Department of Computer Science and Engineering, Indian Institute of technology (IIT), Kharagpur. He has served as the dean of faculty & planning as well as the deputy director of IIT Kharagpur. His research interests include database systems, computer security and multimedia systems. He has published more than two hundred research papers in reputed international journals and conferences. He is a senior member of the IEEE.

JAIDEEP VAIDYA received the bachelor's degree in computer engineering from the University of Mumbai, and the master's and PhD degrees in computer science from Purdue University. He is a professor of computer information systems with Rutgers University. His research interests include privacy, security, and data management. He has published more than 170 papers in international conferences and journals. He is a senior member of the IEEE and an ACM distinguished scientist, and is an editor in chief of the *IEEE Transactions on Dependable and Secure Computing*.

VIJAYALAKSHMI ATLURI is a professor of computer information systems with the MSIS Department, and research director for the Center for Information Management, Integration and Connectivity (CIMIC), Rutgers University. Her research interests include information security, spatial databases, and distributed systems. She has published extensively in premier journals and conferences. She is a senior member of the IEEE Computer Society and a member of the ACM.