On the Complexity of Authorization in RBAC under Qualification and Security Constraints

Yuqing Sun, Qihua Wang, Ninghui Li, *Senior Member*, *IEEE*,
Elisa Bertino, *Fellow*, *IEEE*, and Mikhail (Mike) J. Atallah, *Fellow*, *IEEE*

Abstract—In practice, assigning access permissions to users must satisfy a variety of constraints motivated by business and security requirements. Here, we focus on Role-Based Access Control (RBAC) systems, in which access permissions are assigned to roles and roles are then assigned to users. User-role assignment is subject to role-based constraints, such as mutual exclusion constraints, prerequisite constraints, and role-cardinality constraints. Also, whether a user is qualified for a role depends on whether his/her qualification satisfies the role's requirements. In other words, a role can only be assigned to a certain set of qualified users. In this paper, we study fundamental problems related to access control constraints and user-role assignment, such as determining whether there are conflicts in a set of constraints, verifying whether a user-role assignment satisfies all constraints, and how to generate a valid user-role assignment for a system configuration. Computational complexity results and/or algorithms are given for the problems we consider.

Index Terms—Access control, RBAC, formal methods, computational complexity.

1 Introduction

Role-based access control (RBAC) has established itself as a well-accepted model for access control in many organizations and enterprises. The notion of roles adds a level of indirection to simplify the management of the many-to-many relation between users and permissions. Many companies and institutes with a large number of users and different security requirements are using or considering migrating to RBAC systems.

An advantage of RBAC is that one can specify constraints to enforce higher level security objectives. For instance, mutually exclusive role constraints require that a user cannot have more than k roles in a certain set, where k is an integer. They can be used to enforce separation of duty policies [13]. These constraints are known as Static Separation of Duty constraints in the ANSI RBAC standard [3]. Mutually exclusive role constraints are also supported by major real-world RBAC systems, such as IBM Tivoli Identity Manager [10]. In its latest version (v5.1), Tivoli Identity Manager introduces Separation of Duty rules, which are essentially the mutually exclusive role constraints in this paper.

Role-based constraints can also be used to enforce practical restrictions or business requirements in addition to security policies. For example, a role-cardinality constraint may state that role r must be assigned to at least k users. Such a constraint can be used when the company requires k instances of the task represented by role r be performed simultaneously (in this case, at least k members of role r are needed). For another example, prerequisite constraints, which require that a member of a role r must also be a member of some other roles, can be used in cases, where a number of responsibilities are prerequisites for a certain task.

Even though role-based constraints have been widely studied, it is somewhat surprising that little attention has been paid on the interaction among such constraints. Companies and institutes have a variety of security and practical needs, which indicates that different types of constraints may coexist in a system. Those constraints may conflict with each other, which makes it impossible to assign roles to users while satisfying all constraints. For example, a prerequisite constraint in a system may require that any member of role r_1 must be a member of r_2 as well, while there is another constraint in the system stating that r_1 and r_2 are mutually exclusive. In this case, we cannot assign r_1 to users without violating either the prerequisite constraint or the mutually exclusive role constraint. In this paper, we perform a thorough analysis on the consistency problem on different types of role-based constraints.

In RBAC systems, users gain permissions through role memberships. A fundamental problem in RBAC is assigning roles to users in an appropriate manner. Besides role-based constraints, a valid user-role assignment should also meet certain restrictions on users, which are called user-based constraints in this paper. A natural type of user-based constraints is user-qualification constraints. In real world, a user may be asked to perform a task only if she is qualified to do so, and a task or a job responsibility is normally represented as a role in RBAC. A user-qualification constraint requires that a role, which is associated with a qualification requirement, can only be assigned to those users who satisfy the requirement. For example, we may have a role that should be taken only by users with adequate

- *Y. Sun is with the Department of Computer Science and Technology, Shandong University, Shunhua Road, High-Tech Zone, Jinan, Shandong, China 250100. E-mail: sun_yuqing@sdu.edu.cn.*
- *Q. Wang is with the IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95123. E-mail: qwang@us.ibm.com.*
- *N. Li, E. Bertino, and M.J. Atallah are with the Department of Computer Science, Purdue University, 305 N. University Street, West Lafayette, IN 47907. E-mail: {ninghui, bertino, mja}@cs.purdue.edu.*

Manuscript received 10 Nov. 2008; revised 13 Dec. 2009; accepted 6 May 2010; published online 24 Sept. 2010.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-2008-11-0170. Digital Object Identifier no. 10.1109/2010.55.

accreditation credentials as well as more than 500 h training and three years similar work experiences. Another type of user-based constraint is user-capacity constraints, which restrict the number of roles a user can be assigned to. Assigning too many roles to a user may make her overwhelmed and/or increase the chance of frauds.

To the best of our knowledge, we are the first to study the user-role assignment problem in RBAC models with both role-based and user-based constraints. Considering user-based constraints such as user qualifications makes our settings closer to practice. Since large companies can easily have thousands of users and hundreds of roles in their RBAC systems, it is important to understand the complexity of user-role assignment when constraints are present.

Our contributions are summarized as follows:

- We define the problem of user-role assignment under several kinds of constraints motivated by business and security requirements. In our definition, a valid user-role assignment must not violate any role-based or user-based constraint. We consider three types of role-based constraints; they are prerequisite constraint, mutual exclusion constraint, and role-cardinality constraint. These constraints can be verified efficiently, and they can be used to enforce a variety of access control policies, such as separation of duty policies and resiliency policies. User-based constraints include qualification constraint and user-capacity constraint. In particular, a user is qualified for a role only if his/her qualification satisfies the role's requirements. The user-based constraints we define capture practical needs in real world.
- We study the Constraint Consistency Problem (CCP), which asks whether the existing role-based constraints are consistent, i.e., whether it is possible to assign every role to at least one user while satisfying all given role-based constraints. We show that CCP is **NP**-complete, in general, with respect to the size of roles and the number of constraints. To better understand how different kinds of role-based constraints may affect the complexity of CCP, we study the computational complexities of CCP in different subcases, where only a subset of the three types of role-based constraints are used and/or certain constraints take special forms.
- We study the Assignment Feasibility Problem (AFP), which asks whether there exists a valid user-role assignment under a given configuration with both role-based and user-based constraints. We show that AFP is **NP**-complete with respect to the size of configuration, which consists of a set of users, a set of roles, and a number of constraints. Similar to the study of CCP, we study the computational complexities of AFP in different subcases.
- We study the Assignment Generation Problem (AGP), which returns a valid user-role assignment (if any) for a given configuration. Even though AGP is **NP**-complete, many instances of it may still be efficiently solvable in practice. We present an algorithm for AGP. Our algorithm takes advantages of the observation that AGP can be efficiently formulated as the Boolean satisfiability problem (SAT). This enables us to employ existing SAT solvers to solve the problem and benefit from several decades of research in designing SAT solvers.

The rest of the paper is organized as follows: We define role-based constraints and study their consistency in Section 2. Then, we introduce user-based constraints and study the Assignment Verification Problem in Section 3. We study the Assignment Feasibility Problem and the Assignment Generation Problem in Sections 4 and 5, respectively. We discuss a possible extension to our definition of the assignment problem in Section 6. Finally, we discuss related work in Section 7 and conclude in Section 8.

2 Role-Based Constraints and Their Consistency

In this section, we introduce constraints on roles and discuss the consistency problem among these constraints. We consider three types of role-based constraints. They are role-cardinality, prerequisite, and mutual exclusion constraints. These constraints (or their special forms) have been considered in existing literature [16], [7], [13], [5].

Role-cardinality(RC)**constraints.** A role-cardinality constraint is represented as $\mathrm{RC}(r, c_l, c_u)$, where r is a role, and $c_l, c_u \in [0, \infty)(c_l \leq c_u)$ are called the *lower bound* and the *upper bound* of role r, respectively.

A cardinality constraint $\mathrm{RC}(r, c_l, c_u)$ is satisfied if and only if the role r is assigned to at least c_l users and no more than c_u users. In practice, we require a role be assigned to at least a certain number of users so as to meet workload or resiliency requirements. In contrast, the upper bound in a constraint makes sure that the role is not assigned to too many users due to resource restrictions or to comply with the principle of least privileges. When $c_u = \infty$, there is no limitation on the maximum number of users assigned to the role. Cardinality constraints were suggested in the influential RBAC96 paper [17].

Prerequisite (PRE) **constraints.** A prerequisite constraint is represented as $\mathrm{PRE}(\mathrm{cond}, r)$, where r is a role, and cond is called the *prerequisite condition* of r and it is an expression consisting of roles, conjunctive operator $\wedge$, and disjunctive operator $\vee$.

A prerequisite constraint $\mathrm{PRE}\ (\mathrm{cond}, r)$ is satisfied if and only if for any member u of role r, the role membership of u satisfies cond. For example, $\mathrm{PRE}(r_1 \wedge r_2, r_3)$ is satisfied if and only if any member of r_3 is also a member of roles r_1 and r_2. Without loss of generality, we assume that every role has at most one PRE constraint.

Prerequisite constraints state that if a user takes a certain responsibility, she is also required to take some other responsibilities. In particular, role hierarchy can be represented and enforced using prerequisite constraints. For example, assume that r_1 is senior to r_2, which is, in turn, senior to r_3 and r_4. We can use two prerequisite constraints to represent such a hierarchy; they are $\mathrm{PRE}(r_2, r_1)$ and $\mathrm{PRE}(r_3 \wedge r_4, r_2)$. With the two constraints, anyone who is a member of the "senior" roles must also be members of those "junior" roles (but not the other way around). In the rest of the paper, we do not explicitly discuss role hierarchy. But

whenever we consider prerequisite constraints, our results apply to cases with role hierarchy as well.

Mutual exclusion(MER)constraints. A mutual exclusion constraint is represented as $\mathrm{MER}(R, k)$, where R is a set of roles and $k \in [2, |R|]$ is an integer ($|R|$ is the number of roles in R).

A mutual exclusion constraint $\mathrm{MER}(R, k)$ is satisfied if and only if no user is assigned to k or more roles in R. When $k = 2$, this constraint indicates that a user can be assigned to at most one role in R (i.e., roles in R are mutually exclusive with each other). MER constraints are often used to enforce separation of duty policies [13], [5]. Furthermore, IBM Tivoli Identity Manager supports MER constraints under the name of separation of duty rules [10].

2.1 The Constraint Consistency Problem

We may have different types of constraints in a system and sometimes it is impossible to satisfy all of them. In those cases, we say that the constraints *conflict* with each other. The following two examples illustrate two cases of conflicts:

Example 1. Role-cardinality constraints may conflict with prerequisite constraints. When r_1 is the prerequisite of r_2 (which may be because of a role hierarchy relationship), then the number of users having r_1 must be greater than or equal to the number of users having r_2. If the role-cardinality constraints have the lower bound of r_2 being greater than the upper bound of r_1, then a conflict occurs. The following is such an example:

$$C_1 = \{\mathrm{RC}(r_1, 1, 1), \mathrm{RC}(r_2, 2, 2), \mathrm{PRE}(r_1, r_2)\}.$$

The constraint $\mathrm{RC}(r_2, 2, 2)$ requires r_2 be assigned to exactly two users. According to $\mathrm{PRE}(r_1, r_2)$, any member of r_2 must be a member of r_1. Hence, the two members of r_2 are also members of r_1, which indicates that there are at least two members of r_1. This violates that constraint $\mathrm{RC}(r_1, 1, 1)$, which states that r_1 can be assigned to only one user.

Example 2. Mutual exclusion constraints may conflict with prerequisite constraints. A prerequisite constraint will require certain role memberships to be held together, whereas mutual exclusion constraints prevent certain role memberships to be held together. When they apply to the same set of roles, a conflict occurs. The following is an example:

$$C_2 = \{\mathrm{RC}(r_3, 1, \infty), \mathrm{MER}(\{r_1, r_2\}, 2), \mathrm{PRE}(r_1 \wedge r_2, r_3)\}.$$

The constraint $\mathrm{RC}(r_3, 1, \infty)$ requires r_3 be assigned to at least one user. According to $\mathrm{PRE}(r_1 \wedge r_2, r_3)$, any member of r_3 must be a member of both r_1 and r_2. However, $\mathrm{MER}(\{r_1, r_2\}, 2)$ requires that no user can be a member of both r_1 and r_2. Therefore, it is not possible to satisfy all the three constraints in C_2.

Determining whether there are conflicts in a set of constraints is a fundamental problem and we will study this problem in this section. When conflicts are detected in the constraints, there are a couple of potential approaches to resolve such conflicts. One way is to ask administrators to manually remove a constraint from each conflicting pairs. Another way is to assign priorities to constraints so that when conflicts occur between two constraints, the one with higher priority will automatically override the one with lower priority. Detailed discussion on the strategies for the resolution of conflicts is beyond the scope of this paper.

Definition 1 (Consistency). *Given a set of roles R, we say that a set C of constraints are* consistent *if and only if there exist a set U of users and a user-role assignment $UR \subseteq U \times R$ such that every role in R is assigned to at least one user in UR and no constraint in C is violated by UR.*

The problem of determining whether a set of constraints are consistent is called the CCP.

In the above definition, we require that every role in R is assigned to at least one user. This can be viewed as having an RC constraint for each role in R such that the lower bound of any role is at least one. Without such a requirement, any CCP instance is trivially true: a user-role assignment may simply not assign any user to those roles appearing in conflicting constraints; in other words, for any pair of constraints c_1 and c_2 in C, if c_1 and c_2 cannot be satisfied at the same time, the assignment keeps the roles appearing in c_1 and c_2 empty.

In the CCP problem, we are considering whether a set of constraints are satisfiable with the flexibility of using as many users as one wants to. This is about whether a set of role-based constraints are fundamentally conflicting. Even when they are consistent, there may not be a way to assign all roles to users when the set of users and their qualifications and capacities are fixed. The feasibility of assigning roles to a fixed set of users will be studied in Section 4.

In the following, we study the computational complexity of CCP. The constraints we consider are the three types of constraints (i.e., RC, PRE, and MER) introduced earlier in this section. In the most general case, all three types of constraints are used. But in practice, we sometimes only use a subset of the three types of constraints, and we sometimes use only the limited forms of these types of constraints, making the consistency problem easier to solve. These limited forms are given below.

- RC constraints that do not have upper bound requirements (i.e., $c_u = \infty$), represented as $\mathrm{RC} : \mathit{lower}$.
- MER constraints with $k = 2$, represented as $\mathrm{MER} : 2$.
- PRE constraints whose prerequisite condition only uses conjunctive operator $\wedge$, represented as $\mathrm{PRE} : \mathit{conj}$

  As stated earlier, role hierarchy can be encoded with PRE constraints that use conjunction only.

To represent a subcase of CCP, we list the constraints that can be used in the subcase and whether they are in special form within a pair of braces. For example, $\mathrm{CCP}\langle \mathrm{RC} : \mathit{lower} + \mathrm{MER}\rangle$ denotes the subcase, where only RC constraints with lower bound requirements and mutual exclusion constraints are allowed. $\mathrm{CCP}\langle \mathrm{RC} + \mathrm{MER} + \mathrm{PRE}\rangle$ is the most general case. Note that all the subcases of CCP have RC constraints with $c_l \geq 1$ for every role, because CCP requires each role be assigned to at least one user.

Theorem 1. *The computational complexities of CCP and its subcases are given in Fig. 1.*

<table>
<tr><th></th><th colspan="3">RC constraints: lower bound only</th><th colspan="3">RC constraints</th></tr>
<tr><th></th><th>No MER</th><th>MER: k = 2</th><th>MER</th><th>No MER</th><th>MER: k = 2</th><th>MER</th></tr>
<tr><td>No PRE</td><td colspan="6">A.C. (Always Consistent)</td></tr>
<tr><td>PRE: Conj only
Or with RH</td><td rowspan="2">A.C.</td><td colspan="2">In P</td><td>In P</td><td colspan="2" rowspan="2">NPC
Graph coloring</td></tr>
<tr><td>PRE</td><td colspan="2">NPC
Graph coloring</td><td>NPC
Set Covering</td></tr>
</table>

Fig. 1. Computational complexities of different subcases of CCP.

From Fig. 1, we can see that certain types of constraints do no conflict with each other. For those combinations of constraints that may conflict, we have given the computational complexity of CCP. The bottom-right cell in the table represents the most general case of CCP.

The proof of Theorem 1 consists of four parts. First, we show that certain combinations of constraints cannot have conflicts. Second, we show that CCP is in **NP** in general. Third, we prove that $\mathrm{CCP}\ \langle \mathrm{RC} : \mathit{lower} + \mathrm{PRE} : \mathit{conj} + \mathrm{MER}\rangle$ and $\mathrm{CCP}\ \langle \mathrm{RC} + \mathrm{PRE} : \mathit{conj}\rangle$ are in **P**. Finally, we show that $\mathrm{CCP}\ \langle \mathrm{RC} : \mathit{lower} + \mathrm{PRE} + \mathrm{MER} : 2\rangle, \mathrm{CCP}\ \langle \mathrm{RC} + \mathrm{PRE}\rangle$, and $\mathrm{CCP}\ \langle \mathrm{RC} + \mathrm{PRE} : \mathit{conj} + \mathrm{MER} : 2\rangle$ are **NP**-hard. Other results in Fig. 1 can be implied from the proved cases. In the rest of this section, we provide the first three parts of the proof of Theorem 1; the proofs to the intractability results in Theorem 1 are given in Appendix A.

First, we prove the always-consistent cases in Fig. 1.

Lemma 2. *The answer to CCP is always "yes" when there is no PRE constraint.*

Proof. When there is no PRE constraint, roles can be assigned independently, i.e., assigning one role to a user does not require any additional assignments of other roles to the same user. Hence, different roles can be assigned to different users without violating MER constraints. More specifically, we can construct a user-role assignment UR such that every user has exactly one role and for every RC constraint $\mathrm{RC}(c_l, c_u, r), r$ is assigned to c_l users. In this case, every RC constraint is satisfied, and since every user has only one role, no MER constraint is violated. □

We note that since role hierarchies are encoded as PRE constraints, the above result does not apply to RBAC systems with role hierarchies.

Lemma 3. *The answer to CCP is always "yes" when there is no MER constraint and all the RC constraints have lower bound requirements only.*

Proof. When there is no MER constraint, we may assign as many roles to a user as needed to satisfy PRE constraints. In the extreme case, we assign all roles to a user, and as long as we have enough such users, the lower bound requirements in the RC constraints will be satisfied.

More specifically, let x be the largest value of the lower bounds among the RC constraints. We create x users and assign all the roles in R to each of them. Since the RC constraints have no upper bound requirement and every role is assigned to x users (which is no smaller than any lower bound), all the RC constraints are satisfied. Also, every prerequisite condition is satisfied, because every user is a member of all the roles appearing in the condition, and thus, the user must satisfy the condition. □

Next, we prove that CCP is in **NP** in general and two of its subcases are in **P**.

Lemma 4. *CCP is in* **NP**.

Proof. A nondeterministic Turing machine may guess a UR, and then, verify whether every role is assigned to at least one user and whether all constraints are satisfied. The size of UR is bounded by $|U| \times |R|$. It is clear that determining whether a UR violates a MER , PRE, or RC constraint can be done in polynomial time. Hence, CCP is in **NP**. □

Lemma 5. $\mathrm{CCP}\ \langle \mathrm{RC} : \mathrm{lower} + \mathrm{PRE} : \mathrm{conj} + \mathrm{MER}\rangle$ *is in* **P**. *In other words, CCP is in* **P**, *if all RC constraints only have lower bound requirements, the prerequisite conditions of all PRE constraints use conjunctive operators only, and MER constraints may take general form.*

Proof. First, we need to check whether the PRE constraints conflict with the MER constraints. If there is no conflict between the two types of constraints, then for every role $r_i (i \in [1, n])$ with prerequisite requirements, we can assign r_i and its prerequisite roles to a user u_i. We may then create a number of users identical to u_i for every $i \in [1, n]$, so as to satisfy lower bound requirements in RC constraints.

More specifically, to determine if the answer to such a CCP instance is "yes," we construct a user-role assignment UR in the following way.

For every PRE constraint $\mathrm{PRE}(\mathrm{cond}, r_x)$, we compute the set S_x of roles any member u of r_x must have. Let R_x be the set of roles appearing in cond. Since cond uses only conjunctive operators, all the roles in R_x must be assigned to u. We call R_x the set of prerequisite roles for r_x. We add R_x and r_x to S_x. Also, note that a role r_i in R_x may also have a PRE constraint, and thus, the set R_i of prerequisite roles of r_i must be added into S_x as well. We do this recursively, until S_x no longer grows. For every MER constraint $\mathrm{MER}(A, k)$, we compute the intersection of S_x and A. Let t be the size of the intersection. In order to assign r_x to a user, we have to assign at least t roles in A to the user. If $t \geq k$, there is no way for us to assign r_x to a user without violating either $\mathrm{PRE}(\mathrm{cond}, r_x)$ or $\mathrm{MER}(A, k)$. Hence, the answer to the CCP instance is "no." On the contrary, if for every MER constraint we have $t < k$, we create a user u_x and assign the set S_x of roles to u_x.

Next, if we finish processing all PRE constraints without answering "no" to the CCP instance, then for every role r_i that has not been assigned to any user (r_i does not have a PRE constraint in this case), we create a fresh user u_i and assign r_i to u_i. In this case, every role has been assigned to at least one user and no MER constraint is violated.

Finally, let c_m be the largest value of the lower bounds in all the RC constraints. We make c_m copies of all the users that have been created. In this case, all the RC constraints are satisfied. And since the original user assignment does not violate any PRE or MER constraint, UR does not violate any constraint after the copies are made. The answer to the CCP instance is "yes." □

Lemma 6. CCP $\langle \mathrm{RC} + \mathrm{PRE} : \mathrm{conj} \rangle$ *is in* **P**. *In other words, CCP is in* **P**, *if no MER constraint is used, the prerequisite conditions of all PRE constraints use conjunctive operators only, and RC constraints may take general form.*

Proof. When there is no MER constraints, a user can have as many roles as needed to satisfy PRE constraints. We just need to check if there exists a role r, such that the lower bound of r is larger than the upper bound of one of its prerequisite roles. If such a role exists, it is not possible to satisfy all constraints. Otherwise, the answer to the CCP instance is "yes."

Similar to the proof to Lemma 5, for every PRE constraint $\mathrm{PRE}(\mathrm{cond}, r_x)$, we recursively compute the set S'_x of prerequisite roles of r_x. We then check if the upper bound of any role in S'_x is smaller than the lower bound of the role r_x. Since only conjunction is used in cond, every role in S'_x is required for a user to be assigned to r_x. For every role $r_i \in S'_x$, let c_{u_i} be the upper bound of r_i. Since there can be at most c_{u_i} members of r_i, there can be no more than c_{u_i} members of r_x. If there exists a role $r_j \in S'_x$, such that $c_{u_j} < c_{l_x}$, where c_{l_x} is the lower bound of r_x, then it is impossible to meet the lower bound of r_x without violating the upper bound requirement of r_j. In this case, the answer to the CCP instance is "no." Otherwise, if such a role r_j does not exist for any PRE constraint, then we can construct a user-role assignment UR as follows:

Without loss of generality, assume that $[r_1, r_2, \ldots, r_n]$ is a ranked list in descending order of the values of lower bounds of roles. Let l_i be the lower bound of r_i, where $i \in [1, n]$. We have $l_1 \geq l_2 \geq \cdots \geq l_n$. We create l_1 users $u_1, \ldots, u_{l_1}$. Next, starting from $i = 1$ to $i = n$, we do the following:

If r_i has not been assigned to any user, we assign r_i to $u_1, \ldots, u_{l_i}$. For every $r_j \in S'_i$ (recall that S'_i is the set of prerequisite roles of r_i), if r_j has not been assigned to any user yet (in this case, we must have $j > i$), we also assign r_j to $u_1, \ldots, u_{l_i}$, and according to the assumption, the upper bound of r_j is no less than l_i. Also, $j > i$ implies that $l_j \leq l_i$. Hence, the lower bound requirement of r_j has been satisfied by our assignment.

In the above construction, we have assigned every role r_i to at least l_i users and no upper bound requirements is violated, and all PRE constraints are satisfied. Therefore, the answer to the CCP instance is "yes." □

3 The User-Based Constraints

In the last section, we defined a number of role-based constraints, and studied the problem of checking whether a set of role-based constraints are consistent. In RBAC, a role normally represents a task or a job responsibility, and roles must be assigned to human users so as to be useful. In practice, not every user is qualified to perform every task and there is a limit on how much work a user can/should perform. In this section, we define such practical restrictions on assigning roles to human users as user-based constraints. A valid user-role assignment must satisfy the role-based constraints as well as the user-based constraints.

User-role qualification (URQ) **relation.** In practice, every job or task responsibility represented by a role may have qualification requirements. A user can be assigned to a role only if she meets the qualification requirements of the role.

For instance, we may assume that every human user in the system has a set of qualification attributes. Examples of qualification attributes are diploma, citizenship, training specialty, training period, certification, etc. In a system configuration, all attributes of a user are assigned values from corresponding domains. Every role has a qualification requirement on the attributes of its members. A qualification requirement may be represented as an expression consisting of user attributes and operators in $\{\neg, \vee, \wedge\}$. A term can be in the form of $(a_i \ \mathrm{op} \ c)$ or $a_i \in S$, where a_i is an attribute, $op \in \{=, \neq, <, >, \leq, \geq\}$, c is a constant value, and S is a set of constant values. The following are examples of qualification requirements on roles:

- $(Degree \geq \text{“Bachelor”}) \wedge (WorkExperience \geq 3 \text{ years})$
- $Specialty \in \{\text{“OSsecurity”}, \text{“DBsecurity”}\}$
- $(Location = \text{“NewYorkCity”}) \wedge \neg(Position \in \{\text{“Manager”}, \text{“Director”}\})$

We say that a user u is *qualified* for a role r if and only if the attributes of u satisfy the qualification requirements of r. We introduce the notion of user-role qualification relation to represent the information about which users are qualified for which roles.

Definition 2 (User-Role Qualification Relation). *Given a set of users U and a set of roles R, $URQ = \{(u, r) \mid (u \ is \ qualified \ for \ r) \wedge u \in U \wedge r \in R\}$ is called the* user-role qualification relation.

We would like to point out that user-role qualification relation is different from user-role assignment. $(u, r) \in URQ$ only implies that u is qualified to be assigned to r, but it is possible that u is never assigned to r due to reasons, such as security constraints or workload consideration.

Also, we emphasize that the results presented in the rest of the paper do not depend on the concrete way on how user attributes and role qualification requirements are specified, nor do the results depend on how requirement satisfaction is determined. The examples given earlier in this section represent a potential way to specify user attributes and role qualification requirements, which aims to give the readers a better understanding on how user-role qualification relation might be specified and computed in practice. In the rest of the paper, we abstract away such details and assume that the user-role qualification relation is given in the configuration.

User-capacity (UC)**constraints.** A user-capacity constraint is represented as $\mathrm{UC}(u, c)$, where u is a user, and $c \in [0, \infty)$ is called the *capacity* of u.

A UC constraint $\mathrm{UC}(u, c)$ is satisfied if and only if u is assigned to no more than c roles. Different users may have different capacities. For instance, a full-time employee is able to take more responsibility than a half-time employee. When $c = \infty$, there is no restriction on the number of roles u may be a member of.

There are a couple of reasons why people may want to restrict the maximum number of roles a user is assigned to. The first one is security concern. If a user is assigned to too many roles, the consequence would be severe if she abuses her privileges. Second, there is a limit on the workload that a human user may assume. Giving too much responsibility to a user may make certain tasks end up unfinished.

Finally, we would like to point out that it is possible to extend UC constraints to a more general form. In the above definition, we implicitly assume that every role has the same risk and/or workload, which may not be the case in certain occasions. A potential extension is to associate a weight to each role and require that the sum of the weights of the roles assigned to a user cannot exceed a certain threshold (i.e., the user's capacity). Such an extension will be discussed in Section 6.

3.1 The Assignment Verification Problem

We have introduced both role-based and user-based constraints. A natural problem that arises is to check whether a user-role assignment satisfies all the constraints in the system. In this section, we formally define and study such a problem.

To begin with, we define the configuration of a system.

Definition 3 (Configuration). *A* configuration *is given as a tuple $\langle U, R, C, URQ\rangle$, where U is a set of users, R is a set of roles, $URQ \subseteq U \times R$ defines a user-role qualification relation. C is a set of constraints and each constraint takes one of the form in set $\{\mathrm{MER}(R_s, k), \mathrm{RC}(r, c_l, c_u), \mathrm{PRE}(\mathrm{cond}, r), \mathrm{UC}(u, c)\}$, where $R_s \subseteq R$, $r \in R$, $u \in U$, $1 \leq c_l \leq c_u$, $c \geq 0$, and $k > 1$.*

In the above definition, user-role qualification relation URQ is given separately from other constraints in C. We do so because URQ is a relation determined by the qualification requirements of roles and the attributes of users, while constraints in C are requirements specified by administrators.

Definition 4 (Valid User-Role Assignment). *Given a configuration $\langle U, R, C, URQ\rangle$ and a user-role assignment $UR \subseteq U \times R$, we say that UR is* valid *under $\langle U, R, C, URQ\rangle$ if and only if all the following are true:*

- *Every role $r \in R$ is assigned to at least one user.*
- *$UR \subseteq URQ$. In other words, every role is assigned only to qualified users.*
- *No constraint in C is violated.*

Similar to Definition 1, in Definition 4, we require that every role must be assigned to at least one user so that the tasks represented by the role can be performed.

Definition 5 (AVP). *Given a configuration $\langle U, R, C, URQ\rangle$ and a user-role assignment UR, the* Assignment Verification Problem *(AVP) determines whether UR is valid under $\langle U, R, C, URQ\rangle$.*

The following theorem states that AVP is in **P**:

Theorem 7. *AVP can be solved in quadratic time.*

Proof. To determine whether the given user-role assignment UR is valid under $\langle U, R, C, URQ\rangle$, we need to check three things: 1) whether every role is assigned to at least one user; 2) whether the assignment is compliant with the qualification relation (i.e., $UR \subseteq URQ$); and 3) no constraint in C is violated. It is clear that the first one can be checked in linear time and the second can be checked in quadratic time. And according to the definitions of PRE, MER , RC, and UC constraints, checking whether a constraint is satisfied by a user-role assignment can be done in linear time as well. There are no more than n constraints, where n is the size of input. Therefore, AVP can be solved in quadratic time. □

4 The Assignment Feasibility Problem

Given a configuration $\langle U, R, C, URQ\rangle$, our ultimate goal is to find a valid user-role assignment for $\langle U, R, C, URQ\rangle$. However, not every configuration has a valid assignment. For example, if *Alice* is the only user who is qualified for roles r_1 and r_2 and there is a constraint $\mathrm{MER}(r_1, r_2)$, then there is no way to assign both r_1 and r_2 to a qualified user without violating the mutual exclusion constraint. Configurations, not having any valid user-role assignment, are probably not what designers desire. The AFP problem discussed in this section performs a sanity check on a given configuration and answers the fundamental problem of whether there exists a feasible solution.

Definition 6 (AFP). *Given a configuration $\langle U, R, C, URQ\rangle$, the AFP is to determine whether there exists a valid user-role assignment under $\langle U, R, C, URQ\rangle$.*

Next, we study the computational complexity of AFP. We will show that AFP in the most general case (i.e., with all four types of constraints) is **NP**-complete. Similar to the study of CCP in Section 2, in order to understand how different types of constraints affect the computational complexity of AFP, we consider all subcases, where only a subset of the four types of constraints in C are allowed. Furthermore, RC, PRE, and MER constraints can take special forms. We will study these special cases as well. Note that we always have RC constraints with lower bound requirements in the subcases, as AFP requires that every role is assigned to at least one user, which can be viewed as every role has a lower bound that is at least one.

Also, AFP is at least as difficult as CCP, because subcases of CCP can be reduced to corresponding subcases of AFP. When reducing a CCP instance to an AFP instance, we just need to have enough number of users in the configuration and make every user qualified for every role in the AFP instance.

Theorem 8. *The computational complexities of AFP and its subcases are given in Fig. 2.*

The proof of Theorem 8 is done in three parts. First of all, we show that AFP is in **NP** in general. Second, we prove that AFP $\langle \mathrm{RC} : \mathit{lower} + \mathrm{PRE}\rangle$ and AFP $\langle \mathrm{RC} + \mathrm{UC}\rangle$ are in **P**. Finally, we show that AFP $\langle \mathrm{RC} : \mathit{lower} + \mathrm{UC} + \mathrm{PRE} : \mathit{conj}\rangle$, AFP $\langle \mathrm{RC} : \mathit{lower} + \mathrm{MER} : 2\rangle$, and AFP $\langle \mathrm{RC} + \mathrm{PRE} : \mathit{conj}\rangle$ are **NP**-hard. Other results in Fig. 2 can be implied from the proved cases. In the rest of this section, we present the first two parts of the proof of Theorem 8. The proof of the intractable results of Theorem 8 is given in Appendix B

	RC constraints: lower bound only			RC constraints		
	No MER	MER: k = 2	MER	No MER	MER: k = 2	MER
No Pre	P	NP-hard (SAT)		P	NP-hard (SAT)	
Pre: Conj				NP-hard (Set Covering)		
Pre						
UC				P		
UC + Pre:Conj	NP-hard (Bin-Packing)			NP-hard (Bin-Packing)		
UC + Pre						

Fig. 2. Computational complexities of different subcases of AFP.

Lemma 9. *AFP is in* **NP**.

Proof. Given a configuration $\langle U, R, C, URQ\rangle$, a nondeterministic Turing Machine can guess an assignment UR and verify whether UR is a valid assignment. The size of UR is bounded by $|U| \times |R|$, and according to Theorem 7, verifying whether UR is a valid assignment can be done in polynomial time. Therefore, the problem is in **NP**. □

Next, we prove the two polynomial-time solvable subcases of AFP.

To prove that $\mathrm{AFP}\langle \mathrm{UC} + \mathrm{RC}\rangle$ is in **P**, we reduce the problem to the MAXIMUM FLOW problem. The MAXIMUM FLOW problem is to find a feasible flow through a single-source, single-sink flow network that is maximum. A flow is feasible if it does not exceed the capacity on each edge of the flow network, and the total amount of incoming flows is equivalent to the total amount of outgoing flows on every node of the network except source and sink. The incoming flow of the source and outgoing of the sink is zero, while the outgoing flow of the source and incoming flow of the sink are equivalent to the flow. MAXIMUM FLOW is well solved and many polynomial time algorithms have been proposed for it, such as the Ford-Fulkerson algorithm [6].

Lemma 10. $\mathrm{AFP}\langle \mathrm{UC} + \mathrm{RC}\rangle$ *is in* **P**.

Proof. We reduce the problem to MAXIMUM FLOW, which is in **P**. Given a configuration $\langle U, R, C, URQ\rangle$, we construct a flow network N in the following way (see also Fig. 3):

- For each $r_i \in R$, we create a node a_i. There is an edge between the source s and a_i. The capacity of the edge is 1 if there is no (explicit) role-cardinality constraint on r_i; otherwise, if there is a constraint $\mathrm{RC}(r_i, c_l, c_h)$, the capacity of the edge is c_l, which is denoted as $RC_l(r_i)$ in Fig. 3.
- For each $u_j \in U$, we create a node b_j. There is an edge between b_j and the sink t. The capacity of the edge is infinite if there is no user-capacity constraint on u_j; otherwise, if there is a constraint $\mathrm{UC}(u_j, c)$, the capacity of the edge is c, which is denoted as $UC(u_j)$ in Fig. 3.
- There is an edge between a_i and b_j if and only if $(u_j, r_i) \in URQ$. The capacity of the edge is 1.

Now, we prove that there is a valid assignment under $\langle U, R, C, URQ\rangle$ if and only if there is a flow f from s to t such that $f = \Sigma_{r_i \in R} RC_l(r_i)$.

On the one hand, assume that there is a valid assignment UR under $\langle U, R, C, URQ\rangle$. For every role $r_i \in R$, if there is a constraint $\mathrm{RC}(r_i, c_l, c_h)$ and r_i is assigned to more than c_l users, then we remove some assignments regarding r_i so that r_i is assigned to exactly c_l users in UR; otherwise, if there is no (explicit) role-capacity constraint on r_i and r_i is assigned to more than one user, then we remove some assignments regarding r_i so that r_i is assigned to exactly one user in UR. It is clear that after the modification, UR is still a valid assignment.

Now, we construct a flow f according to UR in the following way:

- For every edge e between b_j and t, there is a flow $f_e = k_j$, where k_j is the number of roles u_j is assigned to.

  Since UR is a valid assignment, if there is a constraint $\mathrm{UC}(u_j, c), u_j$ is assigned no more than c roles in UR; otherwise, if there is no such a constraint, the capacity of e is infinite. Therefore, k_j is no larger than the capacity of e.
- For every edge e between a_i and b_j, if $(u_j, r_i) \in UR$, then there is a flow $f_e = 1$ on e.

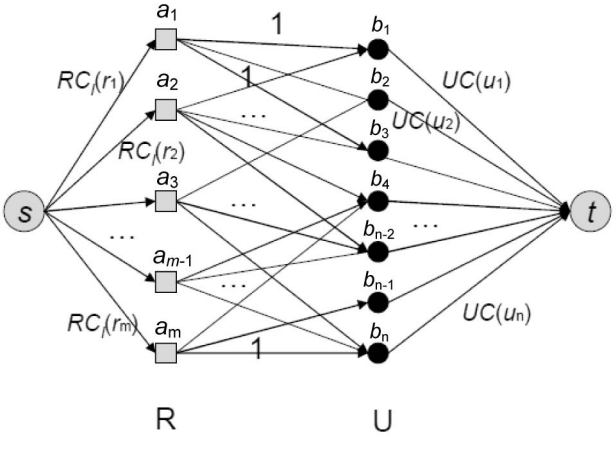


Fig. 3. Reducing $\mathrm{AFP}\langle \mathrm{UC}, \mathrm{RC}\rangle$ to a flow network. $RC_l(r) = c_l$ if there is an explicit RC constraint $\mathrm{RC}(r, c_l, c_u)$; otherwise, $RC_l(r) = 1$. $UC(u) = c$ if there is a constraint $\mathrm{UC}(u, c)$; otherwise, $UC(u) = \infty$.

According to the previous step, the total amount of outgoing flows for b_j is equivalent to the number of roles u_j is assigned to in UR. With the current step, the total amount of incoming flows for b_j is equivalent to the total amount of outgoing flows for b_j.

- For every edge e between s and a_i, there is a flow $f_e = RC_l(r_i)$.

  According to the previous step, the total amount of outgoing flows for a_i is equivalent to the number n_i of users r_i is assigned to in UR. In UR, if there is a constraint $\mathrm{RC}(r_i, c_l, c_h)$, then $n_i = c_l$; otherwise, $n_i = 1$. According to the construction of the flow network, we have $n_i = RC_l(r_i)$, which indicates that the total amount of the incoming flows is equivalent to that of the outgoing flows on a_i.

In general, we have proved that the flow f is a valid flow during our construction. Also, it is easy to see from the last step of the construction that $f = \Sigma_{r_i \in R} RC_l(r_i)$.

On the other hand, assume that there is a flow f from s to t such that $f = \Sigma_{r_i \in R} RC_l(r_i)$. We construct an assignment UR such that $(u_j, r_i) \in UR$ if and only if there is a flow from a_i to b_j in f. Since $f = \Sigma_{r_i \in R} RC_l(r_i)$, it must be the case that every edge between s and a_i is fully loaded. According to the construction of the flow network, if there is a constraint $\mathrm{RC}(r_i, c_l, c_h)$, then $RC_l(r_i) = c_l$, and thus, r_i is assigned to exactly c_l users in UR. Also, if there is a constraint $\mathrm{UC}(u_j, c)$, the capacity of the edge between s and b_j guarantees that the total amount of outgoing flow of b_j is no more than c. Therefore, u_j is assigned to no more than c roles in UR. Finally, according to the construction of the flow network, if there is an edge from a_i to b_j , then $(u_j, r_i) \in URQ$, which indicates that $UR \subseteq URQ$. In general, UR is a valid assignment under $\langle U, R, C, URQ \rangle$. □

Next, we prove that AFP $\langle \mathrm{RC} : \mathrm{lower} + \mathrm{PRE} \rangle$ is in **P**. In the proof to AFP $\langle \mathrm{RC} : \mathit{lower} + \mathrm{PRE} \rangle$, we try to assign as many roles to every user as possible, while the user-role assignment is restricted by user-qualification relation and PRE constraints. We then check whether the lower bound requirements of all roles are satisfied or not.

Lemma 11. AFP $\langle \mathrm{RC} : \mathit{lower} + \mathrm{PRE} \rangle$ *is in* **P**.

Proof. Given a configuration $\langle U, R, C, URQ \rangle$, we try to assign as many roles to every user as possible. To do so, for every user u, we first assign all the roles u is qualified for u, and then revoke those assignments that violate PRE constraints. We then check if the lower bound requirement of any RC constraint is not met.

More specifically, let $u_1, \ldots, u_n$ be the set of users in the configuration. For every user u_i, we compute the maximum set R_i of roles that can be assigned to u_i. R_i is constructed through the following steps:

Step 1: Add all the roles that u_i is qualified for to R_i.

Step 2: For every role $r_j \in R_i$, if r_j has a PRE constraint $\mathrm{PRE}(\mathrm{cond}_j, r_j)$, we check that if the current roles in R_i satisfy cond_j (i.e., we assume that u_i has been assigned all roles currently in R_i and check if u_i satisfies cond_j or not). If R_i does not satisfy cond_j, we remove r_j from R_i, and then, repeat Step 2.

In the above, we construct R_i by first assigning all the roles u_i is qualified for, and then, repeatedly remove those roles, whose prerequisite is not satisfied from R_i until no more role is removed from R_i. In this case, R_i is the maximum set of roles that can be assigned to u_i. We then construct a user-role assignment UR by assigning all the roles in R_i to u_i for every $i \in [1, n]$. It is clear that any valid user-role assignment (if any) must be a subset of UR.

Finally, we check that whether the lower bound requirement of every role is satisfied and whether every role is assigned to at least one user. If the answer is "no," the answer to the AFP instance is "no"; otherwise, the answer to the AFP instance is "yes." □

5 The Assignment Generation Problem

In Section 4, we have studied the AFP, which asks whether a valid user-role assignment exists in a given configuration. A natural question that arises is if valid assignments exist, how can we find one? In this section, we study the AGP, which returns a valid assignment for a given configuration.

First of all, AGP is at least as hard as AFP, because AGP may return an answer if and only if a valid assignment exists. Since the general case of AFP is intractable, AGP is intractable as well. Also, it is not difficult to see that AGP is in **NP**.

Theorem 12. *AGP is* **NP***-complete.*

The fact that AGP is intractable means that there exist difficult problem instances that take exponential time in the worst case. Many instances that will be encountered in practice may still be efficiently solvable. In Section 5.1, we describe an algorithm for AGP.

5.1 An Algorithm for AGP

Our algorithm consists of three parts. First of all, we perform preprocessing to reduce the size and complexity of the given configuration $\langle U, R, C, URQ \rangle$. After this, we reduce the problem (without mutual exclusion, role-cardinality, or user-capacity constraints) to SAT. Finally, we specify Pseudo-Boolean (PB) constraints to handle mutual exclusion, role-cardinality, and user-capacity constraints.

5.1.1 Preprocessing

Given a configuration $\langle U, R, C, URQ \rangle$, we first go through URQ to make sure that every role in R has at least one qualified user. Also, we remove a user from U if he/she is not qualified for any role.

Next, we try to reduce the size of URQ. We design a polynomial-time algorithm that removes (u, r) from URQ if there is a prerequisite constraint $\mathrm{PRE}(\mathrm{cond}, r)$ and it is impossible for u to satisfy cond. The algorithm, which is given in Fig. 4, is very similar to the algorithm used in the proof of AFP$\langle \mathrm{RC} : \mathit{lower} + \mathrm{PRE} \rangle$ in Section 4. For every user u, the algorithm in Fig. 4 computes the maximum set R_u of roles that can be assigned to u, and then, removes (u, r) from URQ for any role $r \notin R_u$.

Finally, we try to reduce the number of constraints or simplify them in the following way:

```
For every user u ∈ U Do
    R_u ← ∅;
    For every role r ∈ R Do
        If (u, r) ∈ URQ Then R_u ← R_u ∪ {r};
    While true Do
        f ← 0;
        For every role r ∈ R_u Do
            If exists PRE(cond, r) and R_u does not satisfy cond Then
                R_u ← R_u − {r};
                URQ ← URQ − {(u, r)};
                f ← 1;
            EndIf;
        If f = 0 Then Break;
    EndWhile;
EndFor;
```

Fig. 4. The algorithm to reduce the size of URQ used in the preprocessing procedure in the algorithm for AGP. For every user u, we compute the maximum set R_u of roles that can be assigned to u and remove (u, r) from URQ for any role $r \notin R_u$.

- For every mutual exclusion constraint $\mathrm{MER}(R', k)$, if there is no user who is qualified for at least k roles in R', then we remove the constraint from C.
- For every role-cardinality constraint $\mathrm{RC}(r, c_l, c_u)$, if less than c_l users are qualified for r, then it is impossible to satisfy the constraint and we return "no answer." Otherwise, if no more than c_u users are qualified for r, we revise that constraint as $\mathrm{RC}(r, c_l, \infty)$. Such a revision will simplify a PB constraint specified in the third part of the algorithm.
- For every user-capacity constraint $\mathrm{UC}(u, c_u)$, if u is qualified for no more than c_u roles, then we remove the constraint from C. Since u can only be assigned to those roles she is qualified for, when u is qualified for less than c_u roles, the constraint $\mathrm{UC}(u, c_u)$ will not be violated by assignments that comply with URQ.

5.1.2 Reduction to SAT

An important observation is that AFP can be efficiently reduced to SAT if we do not consider MER , RC, or UC constraints (i.e., those constraints that have integer parameters). By reducing the problem to SAT, we benefit from several decades of research on the design of SAT solvers. Problems in many fields, including databases, planning, computer-aided design, machine vision, and automated reasoning, have been reduced to SAT and solved using SAT solvers. Oftentimes, this results in better performance than using existing domain-specific algorithms for those problems.

For every $(u_j, r_i) \in URQ$, we specify a variable $v_{j,i}$. That is to say, we have $|URQ|$ variables in our SAT instance. Variable $v_{j,i}$ being set to true indicates that (u_j, r_i) is in the resulting user-role assignment (i.e., r_i is assigned to u_j).

- For every role r_i without an explicit RC constraint: we specify a clause $\phi = \bigvee_{j \in X} v_{j,i}$, where $X = \{j \mid (u_j, r_i) \in URQ\}$.

  The clause states that r_i is assigned to at least one user.
- For every prerequisite constraint $\mathrm{PRE}(\mathrm{cond}, r_i)$: such a constraint essentially states that $r_i \rightarrow \mathrm{cond}$, which can be equivalently written as $\neg r_i \vee \mathrm{cond}$. For every user u_k such that $(u_k, r_i) \in URQ$, we construct a clause $\phi = \neg v_{k,i} \vee g(\mathrm{cond}, k)$, where the function g constructs a clause from cond by replacing every role with a variable: any role r_j in cond is replaced with $v_{k,j}$. For example, $g(r_1 \vee (r_2 \wedge r_3), k) = v_{k,1} \vee (v_{k,2} \wedge v_{k,3})$.

  The clause ϕ states that the role membership of any user who is assigned to r_i must satisfy the precondition cond.

  Note that certain SAT solvers require the input expression be in CNF. In those cases, we will have to revise ϕ into CNF. Revising an expression into CNF could lead to exponential growth in the size of the expression. However, in practice, the precondition cond for a role is normally very simple. Thus, the size expansion of clauses should not be significant in practice.

In general, the resulted Boolean formula for the SAT instance is a conjunction of all the clauses (denoted as ϕ) generated in the above.

5.1.3 Handling MER , RC, and UC Constraints

The reduction to SAT described above does not consider MER , RC, or UC constraints. To handle these three types of constraints, which contain integer parameters, we can use Pseudo-Boolean constraints. In PB constraints, all variables take values of either 0 (false) or 1 (true). Constraints are linear inequalities with integer coefficients, for example, $2v_1 + v_2 + v_3 \geq 2$ is a PB constraint. A disjunctive clause encountered in SAT is a special case of PB constraints; for example, $v_1 \vee v_2 \vee v_3$ is equivalent to $v_1 + v_2 + v_3 \geq 1$. Many SAT solvers also support PB constraints, SAT4J [15], for example.

Given a MER constraint $\mathrm{MER}(R', k)$, let u_j be a user who is qualified for at least k roles in R'. We specify a PB constraint $\Sigma_{r_i \in R'} v_{j,i} < k$. Such a constraint ensures that u_j is assigned to less than k roles in R'.

Also, given a UC constraint $\mathrm{UC}(u_j, c)$, let R_j be the set of roles u_j is qualified for. We specify a PB constraint $\Sigma_{r_i \in R_j} v_{j,i} \leq c$. Such a constraint requires that no more than c variables that are related to u_j may be set to true, which implies that u_j is assigned to at most c roles. Similarly, given an RC constraint $\mathrm{RC}(r_i, c_l, c_u)$, let U_i be the set of users who are qualified for r_i. If c_u is ∞, we specify a PB constraint $\Sigma_{u_j \in U_i} v_{j,i} \geq c_l$; otherwise, we specify two PB constraints, $\Sigma_{u_j \in U_i} v_{j,i} \geq c_l$ and $\Sigma_{u_j \in U_i} v_{j,i} \leq c_u$.

In general, if a truth assignment T is found for the SAT instance with PB constraints, we construct a valid user-role assignment UR for $\langle U, R, C, URQ \rangle$ in the following way:

$$UR = \{(u_j, r_i) \mid (v_{j,i} = \mathrm{true}) \in T\}.$$

Finally, we would like to point out that our algorithm can be easily extended to support the cases where some user-role assignments are fixed. In practice, some user-role assignments may have been predetermined and what we want to do is to find a valid assignment without changing the predetermined assignments. To specify a fixed assignment, say r_i is assigned to u_j, we just need to add a clause $v_{j,i}$ to the SAT formula. Such a clause forces setting $v_{j,i}$ to true in any truth assignment that satisfies the formula.

6 Discussion

In this section, we discuss a possible extension to the user-role assignment problem. As we have mentioned in Section 3,

our current definition of the user-capacity constraint counts every role in the configuration equally. However, in practice, not every role has the same workload. To capture this, we extend our definition to introduce an integer weight to each role. A user-capacity constraint is still represented as $\mathrm{UC}(u, c)$; but now it requires that the sum of the weights of the roles assigned to u must not exceed c. Our original definition on the constraint, which counts the number of roles assigned to a user, is a special case in which every role has weight 1.

Next, we discuss how the extension may affect the results presented in this paper. First of all, it is easy to see that the modified user-capacity constraint can still be verified in polynomial time, as we just need to compute the sum of some integers and do a comparison. This implies that AVP is still in **P** and the general case of AFP is still in **NP**, as a nondeterministic Turing Machine can generate a user-role assignment and verify it in polynomial time. The complexities of those subcases not using user-capacity constraints and those that are **NP**-hard in Fig. 2 remain unchanged. However, $\mathrm{AFP}(\mathrm{RC} : \mathrm{lower} + \mathrm{UC})$ is now **NP**-hard, which implies that $\mathrm{AFP}(\mathrm{RC} + \mathrm{UC})$ is **NP**-hard as well.

Lemma 13. $\mathrm{AFP}(\mathrm{RC} : \mathit{lower} + \mathrm{UC})$ *is* **NP***-hard with weighted roles and the revised user-capacity constraints.*

Proof. We can reduce the **NP**-complete SUBSET SUM problem to $\mathrm{AFP}(\mathrm{RC} : \mathit{lower} + \mathrm{UC})$. In SUBSET SUM, we are given a set $S = \{a_1, \ldots, a_m\}$ of integers and an integer k, and we are asked whether there is a subset of integers of S whose sum is k. Without loss of generality, we assume that $a_i > 0$ for every $i \in [1, m]$. We construct a configuration $\langle U, R, C, URQ\rangle$ as follows:

Let $U = \{u_1, u_2\}$ and $R = \{r_1, \ldots, r_m\}$. In URQ, every user is qualified for every role. Let w_i be the weight of r_i. We have $w_i = a_i$. Intuitively, $r_i \in R$ corresponds to $a_i \in S$. Let $w = \Sigma_{i=1}^{m} w_i$. We specify two user-capacity constraints $\mathrm{UC}(u_1, k)$ and $\mathrm{UC}(u_2, w - k)$.

Now, we prove that there is a valid assignment in $\langle U, R, C, URQ\rangle$ if and only if the answer to the SUBSET SUM problem is "yes." On the one hand, if there is a valid assignment in $\langle U, R, C, URQ\rangle$, then there must exist two sets of roles R_1 and R_2, which are assigned to u_1 and u_2, respectively, and $R_1 \cup R_2 = R$. Due to the user-capacity constraints, $\Sigma_{r_i \in R_1} w_i \leq k$ and $\Sigma_{r_j \in R_2} w_j \leq w - k$. Since $R_1 \cup R_2 = R$, we have $\Sigma_{r_i \in R_1} w_i + \Sigma_{r_j \in R_2} w_j = w$. Therefore, $\Sigma_{r_i \in R_1} w_i = k$, which implies that $\Sigma_{r_i \in R_1} a_i = k$ as $w_i = a_i$.

On the other hand, assume that there exists $S_1 \subseteq S$ such that the sum of the integers in S_1 is equivalent to k. Without loss of generality, assume that $S_1 = \{a_1, \ldots, a_t\}$. We construct an assignment UR by assigning $\{r_1, \ldots, r_t\}$ to u_1 and $R - \{r_1, \ldots, r_t\}$ to u_2. In this case, the sum of the weight of the roles assigned to u_1 is k, while that of the roles assigned to u_2 is $w - k$. Therefore, both of the user-capacity constraints are satisfied and UR is a valid assignment. □

Finally, we point out that our algorithm for AGP described in Section 5.1 can be easily modified to support the weighted roles and the revised user-capacity constraints. All we need to do is to modify the Pseudorandom constraints that are used to enforce user-capacity constraints. More specifically, we revise the constraint $\Sigma_{r_i \in R_k} v_{k,i} \leq c$ into $\Sigma_{r_i \in R_k} w_i v_{k,i} \leq c$, where w_i is the weight of r_i.

7 Related Work

This paper studies user-role assignment with qualification and security constraints. Constraint specification and enforcement is a well-studied topic. There exists a wealth of literature [1], [2], [7], [9], [11], [12], [18], [19] on constraints in the context of RBAC. Some proposed and classified new kinds of constraints [9], [18]; some proposed new languages for specifying sophisticated constraints [1], [2], [7], [12], [19]; and other studied whether these constraints comply with higher level policy objectives [5], [13]. Most of these constraints are motivated by SoD and are variants of role mutual exclusion constraints, which may declare two roles to be mutually exclusive so that no user can be a member of both roles. However, the existing works do not consider how to assign users to roles so as to satisfy all the security constraints in a system at the same time.

Our role assignment problem is similar to the problem of assigning users to perform different steps in a workflow, while satisfying a number of constraints, which has been studied in [4], [8], [20]. Intuitively, one can map each role in our setting to one step in a workflow, and the problem of assigning users to roles becomes the same as assigning users to steps. The user-role qualification relation in our setting corresponds to the user-step authorization relation. Role mutual exclusion constraints correspond to mutual exclusion constraints among steps in a workflow. However, our work has two main differences. First, we consider different kinds of constraints because of our different motivation. We have role-cardinality and user-capacity constraints, which have not been studied in the workflow literature. The nature of workflow is that each step is performed by a single user, thus role-cardinality constraints seem pointless in a workflow setting. However, user-capacity constraints might be fruitfully added to workflow. On the other hand, the one-user-per-step nature of workflow enables one to consider more sophisticated constraints such as the two users performing the first and the last step must not be in conflict of interest with each other. Second, we introduce new techniques to solve the assignment generation. For the tractable special cases of the assignment generation problem, we reduce the problem to the Maximum Flow problem. For the general case, we reduce it to SAT with Boolean and Pseudo-Boolean constraints. These techniques may be applicable in the workflow settings as well.

Our user-role assignment problem emphasizes that all roles must be assigned to qualified users so that the tasks represented by them can be performed. This has a similar spirit as resiliency policies [14]. A resiliency policy requires that even if a certain number of users are absent, the remaining users must still have enough permissions to complete a certain task. However, [14] only studies whether an existing access control is resilient or not and does not study how to assign users to permission to satisfy resiliency requirements. They did not consider user qualification in the resiliency policies either.

8 Conclusion and Future Work

In this paper, we have studied the user-role assignment problem with consideration of user-role qualification relation and a variety of role-based and user-based constraints. We have studied the consistency problem among three types of role-based constraints, and studied computational problems related to user-role assignment, such as the AVP and the AFP. Furthermore, we have proposed an algorithm to find a valid user-role assignment for a given configuration. Our algorithm takes advantages of the existence of fast SAT solvers that support Pseudo-Boolean constraints.

Open problems. A future direction is to introduce optimization goals into the user-role assignment problem. An interesting optimization objective is to minimize the number of users. That is, given a configuration, what is the smallest number of users in a valid assignment? Such a problem may help us to find out redundant users and improve the utilization of human resources.

Another future work also relates to human resource management. Assume that there is no valid assignment for a given configuration, which indicates that we may either change the configuration or hire more people (i.e., introduce more users into the system). If we would like to add users into the configuration, what kinds of users are needed? What is the minimum cost of adding new users so that a valid assignment exists for the new configuration (assuming that users with different qualification have different costs)?

Finally, in this paper, we assumed that authorization constraints are given. It will be useful to study how to efficiently generate constraints to enforce access control policies. The problem is particularly challenging when we have multiple policies at the same time in the system, and those policies may be of different types (e.g., some of them are separation of duty policies and some are resiliency policies).

Appendix A

Proof of Theorem 1

In this Appendix, we prove the intractable subcases in Theorem 1. We just need to show that CCP $\langle \mathrm{RC} + \mathrm{PRE} \rangle$, CCP $\langle \mathrm{RC} : \mathit{lower} + \mathrm{PRE} + \mathrm{MER} : 2 \rangle$, and CCP $\langle \mathrm{RC} + \mathrm{PRE} : \mathit{conj} + \mathrm{MER} : 2 \rangle$ are **NP**-hard. Other intractability results can be implied from the three cases.

Lemma 14. CCP $\langle \mathrm{RC} + \mathrm{PRE} \rangle$ *is* **NP***-hard. In other words, CCP is* **NP***-hard, if no MER constraint is used and all other constraints may take general forms.*

Proof. We reduce the **NP**-complete `Set Covering` problem to CCP. In the `Set Covering` problem, given a set $S = \{e_1, \ldots, e_m\}$, a family $F = \{S_1, \ldots, S_n\}$ of S's subsets, and an integer k, we are asked whether there exist k elements (which are sets) in F, whose union is S.

Given an instance of the `Set Covering` problem, we construct such a CCP instance: We create m roles, $a_1, \ldots, a_m$, and another n roles, $b_1, \ldots, b_n$. For each $i \in [1, n]$, we specify a constraint $\mathrm{RC}(b_i, 1, 1)$, which requires that b_i must be assigned to exactly one user. Also, we create two other roles x and y, with constraints $\mathrm{RC}(x, 1, k)$ and $\mathrm{RC}(y, n, n)$. In other words, role x can be assigned to at most k users and role y must be assigned to exactly n users. Also, we specify a constraint $\mathrm{PRE}(b_1 \vee \cdots \vee b_n, y)$. Since b_i can be assigned to only one user and y must be assigned to n users due to the RC constraints, the PRE constraint $\mathrm{PRE}(b_1 \vee \cdots \vee b_n, y)$ together with the RC constraints ensures that b_i and b_j, where $i \neq j$, must be assigned to different users. Finally, for every $i \in [1, m]$, let $\{S_{d_{i,1}}, \ldots, S_{d_{i,t}}\}$ be the set of elements in F such that $e_i \in S_{d_{i,j}} (j \in [1, t])$; we specify a constraint $\mathrm{PRE}((b_{d_{i,1}} \vee \cdots \vee b_{d_{i,t}}) \wedge x, a_i)$.

Next, we prove that the answer to the `Set Covering` instance is "yes" if and only if the answer to the CCP instance is "yes."

On the one hand, without loss of generality, assume that the union of $S_1, \ldots, S_k$ is S. We can construct a user-role assignment UR in such a way: we create n users $u_1, \ldots, u_n$. For every $i \in [1, n]$, we assign roles b_i and y to u_i. This satisfies the constraints $\mathrm{RC}(b_i, 1, 1)$, $\mathrm{RC}(y, n, n)$, and $\mathrm{PRE}(b_1 \vee \cdots \vee b_n, y)$. We also assign role x to the k users $u_1, \ldots, u_k$. Finally, we assign role a_i to user u_j if and only if $e_i \in S_j$ and $j \leq k$; this satisfies $\mathrm{PRE}((b_{d_{i,1}} \vee \cdots \vee b_{d_{i,t}}) \wedge x, a_i)$, since u_j has been assigned to x when $j \leq k$. By assumption, for every $i \in [1, m]$, we have $e_i \in (S_1 \vee \cdots \vee S_k)$. Hence, for every $i \in [1, m]$, a_i has been assigned to at least one user in $\{u_1, \ldots, u_k\}$. In general, every role has been assigned to at least one user and no constraint is violated. The answer to the CCP instance is "yes."

On the other hand, assume that we have a user-role assignment UR that is valid with respect to the CCP instance. Due to the constraint $\mathrm{RC}(1, k, x), x$ can be assigned to no more than k users. Without loss of generality, assume that x is assigned to $u_1, \ldots, u_k$. In this case, roles in $\{a_1, \ldots, a_m\}$ can only be assigned to these k users. Also, in order for $u_j (j \leq k)$ to be a member of a_i without violating the constraint $\mathrm{PRE}((b_{d_{i,1}} \vee \cdots \vee b_{d_{i,t}}) \wedge x, a_i), u_j$ must be assigned to a role in $\{b_1, \ldots, b_n\}$. As stated during the construction of the CCP instance, a user can be assigned to at most one role in $\{b_1, \ldots, b_n\}$. Without loss of generality, assume that $u_j (j \leq k)$ is assigned to b_{c_j}, where $c_j \in [1, n]$. Now, we prove that the union of $S_{c_1}, \ldots, S_{c_k}$ is S. For every $i \in [1, m], a_i$ is assigned to at least one user in UR. And we have argued that a_i can only be assigned to users in $\{u_1, \ldots, u_k\}$. Without loss of generality, assume that a_i is assigned to u_1. In this case, in order to satisfy that constraint $\mathrm{PRE}((b_{d_{i,1}} \vee \cdots \vee b_{d_{i,t}}) \wedge x, a_i), b_{c_1}$ must be in the prerequisite condition of a_i, since u_1 is a member of b_{c_1} but no other roles in $\{b_1, \ldots, b_n\}$. According to the construction of the CCP instance, we must have $e_i \in S_{c_1}$. Therefore, the union of $S_{c_1}, \ldots, S_{c_k}$ is S. The answer to the set covering instance is "yes." □

Lemma 15. CCP $\langle \mathrm{RC} : \mathrm{lower} + \mathrm{PRE} + \mathrm{MER} : 2 \rangle$ *is* **NP***-hard. In other words, CCP is* **NP***-hard, if all RC constraints only have lower bound requirements, all the MER constraints have $k = 2$, and PRE constraints may take general form.*

Proof. We reduce the **NP**-complete `Graph K-Coloring` problem to CCP. In the `Graph K-Coloring` problem, given a graph G and a number k, we are asked whether we can assign one of the k colors to every node in G, such that no pair of adjacent nodes are assigned the same color.

Given a graph G and a number k, we construct such a CCP instance: Assume that there are m nodes in G. For

every node n_i in G, we create k roles $r_{i,1}, \ldots, r_{i,k}$. For every pair of adjacent nodes (n_i, n_j), we specify k MER constraints $\mathrm{MER}(\{r_{i,1}, r_{j,1}\}), \ldots, \mathrm{MER}(\{r_{i,k}, r_{j,k}\})$. We then create a role x along with a PRE constraint $\mathrm{PRE}(F_1 \wedge \cdots \wedge F_m, x)$, where $F_i = r_{i,1} \vee \cdots \vee r_{i,k}$.

Next, we prove that G can be colored with k colors validly if and only if the answer to the CCP instance if "yes."

On the one hand, assume that G can be colored with k colors validly. We can construct a user-role assignment UR in such a way based on a valid coloring of G: Create a user u_x and assign role x to her. No other user will be assigned to x. Also, for every node n_i in G, if n_i is assigned the jth color, we assign $r_{i,j}$ to u_x. In this case, every $F_j(j \in [1, m])$ is satisfied, and thus, $\mathrm{PRE}(F_1 \wedge \cdots \wedge F_m, x)$ is satisfied. Since G is colored validly, no pair of adjacent nodes (n_i, n_j) is assigned to the same color, which indicates that no MER constraint is violated by the role assignments for u_x. For every role that is not assigned to u_x, we create a fresh user and assign the role to her. All such users have only one role and thus they do not violate any MER constraints. In this case, the user-role assignment UR meets all requirements, and thus, the answer to the CCP instance is "yes."

On the other hand, assume that the answer to the CCP instance is "yes". There must exist a user-role assignment UR that meets the requirements in the CCP instance. We can color G in such a way: Let u_x be the user, who is assigned role x. For every $i \in [1, m]$, u_x must be assigned to at least one role in $\{r_{i,1}, \ldots, r_{i,k}\}$ due to the PRE constraint of x. For every node n_i in G, if u_x is assigned to $r_{i,j}$, then we assign the jth color to n_i (if n_i has not been colored yet). In this way, since the role assignment for u_x does not violate any MER constraints, no pair of adjacent nodes in G are given the same color. Therefore, we have colored G validly.

In general, G can be colored with k colors validly if and only if the answer to the CCP instance if "yes." The lemma holds. □

Lemma 16. $\mathrm{CCP}\ \langle \mathrm{RC} + \mathrm{PRE} : \mathrm{conj} + \mathrm{MER} : 2 \rangle$ *is* **NP***-hard. In other words, CCP is* **NP***-hard, if all PRE constraints only use conjunction in their conditions, all the MER constraints have $k = 2$, and RC constraints may take general form.*

Proof. We reduce the **NP**-complete `Graph K-Coloring` problem to CCP. In the `Graph K-Coloring` problem, given a graph G and a number k, we are asked whether we can assign one of the k colors to every node in G, such that no pair of adjacent nodes are assigned the same color.

Given a graph G and an integer k, we construct such a CCP instance: Assume that there are m nodes in G. We create a role x with a constraint $\mathrm{RC}(x, 1, k)$. For every node $n_i(i \in [1, m])$, we create a role r_i with a constraint $\mathrm{PRE}(x, r_i)$. For every pair of adjacent nodes (n_i, n_j) in G, we create a constraint $\mathrm{MER}(\{n_i, n_j\}, 2)$.

Next, we prove that G can be colored with k colors validly if and only if the answer to the CCP instance if "yes."

On the one hand, assume that G can be colored with k colors validly. We can construct a user-role assignment UR in such a way based on a valid coloring of G: We create k users $u_1, \ldots, u_k$, and assign role x to these k users. This does not violate the constraint $\mathrm{RC}(x, 1, k)$. For every node $n_i(i \in [1, m])$, if n_i is assigned the jth color, where $j \in [1, k]$, we assign the role r_i to u_j. Since u_j is a member of x, the constraint $\mathrm{PRE}(x, r_i)$ is satisfied. Also, since the coloring is valid, no pair of adjacent nodes are assigned the same color. This indicates that no pair of mutually exclusive roles are assigned to the same user. Hence, no MER constraint is violated. In this case, the user-role assignment UR meets all requirements, and thus, the answer to the CCP instance is "yes."

On the other hand, assume that the answer to the CCP instance is "yes." There must exist a user-role assignment UR that meets the requirements in the CCP instance. We can color G in such a way: Without loss of generality, assume that x is assigned to $u_1, \ldots, u_k$. For every role $r_i(i \in [1, m])$, r_i must be assigned to users in $\{u_1, \ldots, u_k\}$ due to the constraint $\mathrm{PRE}(x, r_i)$. Let $u_j(j \in [1, k])$ be a member of r_i. We assign the jth color to n_i (if n_i has not been colored yet). All the MER constraints being satisfied indicates that no pair of mutually exclusive roles are assigned to the same user. Hence, no pair of adjacent nodes in G are assigned to the same color.

In general, G can be colored with k colors validly if and only if the answer to the CCP instance if "yes." The lemma holds. □

APPENDIX B
PROOF OF THEOREM 8

In this Appendix, we prove the intractable subcases in Theorem 8. We just need to show that $\mathrm{AFP}\ \langle \mathrm{RC} : \mathit{low} + \mathrm{UC} + \mathrm{PRE} : \mathit{conj} \rangle$, $\mathrm{AFP}\ \langle \mathrm{RC} : \mathit{lower} + \mathrm{MER} : 2 \rangle$, and $\mathrm{AFP}\ \langle \mathrm{RC} + \mathrm{PRE} : \mathit{conj} \rangle$ are **NP**-hard. Other intractability results can be implied from the three cases.

Lemma 17. $\mathrm{AFP}\ \langle \mathrm{RC} : \mathit{low} + \mathrm{UC} + \mathrm{PRE} : \mathit{conj} \rangle$ *is* **NP***-hard.*

Proof. We reduce the **NP**-complete BIN PACKING problem to $\mathrm{AFP}\ \langle \mathrm{RC} : \mathit{low} + \mathrm{UC} + \mathrm{PRE} : \mathit{conj} \rangle$. In BIN PACKING, given a set of integers $S = \{a_1, \ldots, a_n\}$, an integer k, and an integer c, we are asked if we can place all the integers into k bins such that the sum of the integers in each bin is no larger than c. BIN PACKING remains **NP**-complete even if the integers in S are represented in unary. In this proof, we assume that the integers in S are represented in unary.

Given a BIN PACKING instance, we construct an AFP instance as follows: For every $a_i \in S$, we construct a_i roles $r_{i,1}, \ldots, r_{i,a_i}$. If $a_i > 1$, we construct a PRE constraint $\mathrm{PRE}(r_{i,1} \wedge \cdots \wedge r_{i,a_i-1}, r_{a_i})$. Also, we construct k users, and every user is qualified to be assigned to every role. Each user can be assigned to at most c roles.

Next, we show that the answer to the BIN PACKING instance is "yes" if and only if the answer to the AFP instance is "yes."

On the one hand, assume that all the integers in S have been placed into k bins and the sum of integers in each bin is no larger than c. For every bin $B_i(i \in [1, k])$, for every $j \in [1, n]$, if a_j has been placed in B_i, we assign roles $r_{j,1}, \ldots, r_{j,a_j}$ to user u_i. This does not violate the PRE constraint $\mathrm{PRE}(r_{j,1} \wedge \cdots \wedge r_{j,a_j-1}, r_{a_j})$. And since the sum of the integers in B_i is no larger than c, u_i is assigned to no more than c roles, which satisfies the capacity

constraint of u_i. Also, since every $a_j \in S$ has been placed in a certain bin, all the roles are assigned to a certain user. In this case, we have constructed a valid user-role assignment and the answer to the AFP instance is "yes."

On the other hand, assume that there is a valid user-role assignment for the AFP instance. We now place the integers in S to bins based on the user-role assignment. For every $i \in [1,k]$, for every $j \in [1,n]$, if r_{a_j} is assigned to u_i, we place integer a_j into the ith bin B_i. Since every $r_{a_j} (j \in [1,n])$ has been assigned to at least one user, we have placed all the integers in S to the bins. Now, we show that there is no bin, whose sum of integers is greater than c. Due to the constraint $\mathrm{PRE}(r_{j,1} \wedge \ldots \wedge r_{j,a_j-1}, r_{a_j})$, the fact that r_{a_j} is assigned to u_i indicates that all the a_j roles in $\{r_{j,1}, \ldots, r_{j,a_j}\}$ must have been assigned to u_i as well. In this case, it is easy to see that the sum of integers in B_i is no larger than the number of roles assigned to u_i. Since u_i can be assigned to at most c roles, the sum of integers in B_i is no larger than c. Therefore, the answer to the BIN PACKING instance is "yes." □

Lemma 18. $\mathrm{AFP}\ \langle \mathrm{RC} : \mathrm{lower} + \mathrm{MER} : 2 \rangle$ *is* **NP***-hard.*

Proof. The intractability of $\mathrm{AFP}\langle \mathrm{RC} : \mathit{lower}, \mathrm{MER} \rangle$ can be implied by the **NP**-completeness of the Workflow Satisfiability Problem proved in [20]. Please refer to Section 7 for detailed discussion. In below, we give our proof of the **NP**-hardness of $\mathrm{AFP}\langle \mathrm{RC} : \mathit{lower}, \mathrm{MER} \rangle$, which employs a different reduction from the one used in [20].

We reduce the **NP**-complete SAT problem to $\mathrm{AFP}\langle \mathrm{RC} : \mathit{lower}, \mathrm{MER} \rangle$. In SAT, we are given an expression ϕ in conjunctive normal form (CNF) and are asked whether there exists a truth assignment for variables appeared in ϕ such that ϕ is evaluated to true.

Let $\phi = \phi_1 \wedge \cdots \wedge \phi_m$, where $\phi_i = l_{i_1} \vee \cdots \vee l_{i_t}$ is a clause and l_{i_j} is a literal (i.e., a variable or the negation of a variable). Without loss of generality, assume that no clause contains both v and $\neg v$. Let $\{v_1, \ldots, v_n\}$ be the set of variables appeared in ϕ. We construct a configuration $\langle U, R, C, URQ \rangle$ as follows:

Let $U = \{u', u_1, \ldots, u_n\}$ and $R = R_a \cup R_b$, where $R_a = \{a_{1,0}, a_{1,1}, \ldots, a_{n,0}, a_{n,1}\}$ and $R_b = \{b_1, \ldots, b_m\}$. Intuitively, $u_i (i \in [1,n])$ corresponds to variable v_i in the SAT instance; $a_{i,0}$ and $a_{i,1}$ correspond to setting variable v_i to false and true, respectively; and b_j corresponds to clause ϕ_j in the SAT instance. Next, we construct URQ in such a way that: 1) u' is qualified for every role in R_a; 2) $u_i (i \in [1,n])$ is qualified for $a_{i,0}$ and $a_{i,1}$ but not any other role in R_a; and 3) $u_i (i \in [1,n])$ is qualified for b_j if and only if variable v_i or its negation appears in the clause ϕ_j. Finally, for every $i \in [1,n]$, we specify a mutual exclusion constraint $\mathrm{MER}(a_{i,0}, a_{i,1})$, which indicates that variable v_i cannot be set to both false and true. Also, we specify a constraint $\mathrm{MER}(a_{i,0}, b_j)$ if and and only if v_i appears in ϕ_j; we specify a constraint $\mathrm{MER}(a_{i,1}, b_j)$ if and only if $\neg v_i$ appears in ϕ_j.

Now, we prove that ϕ is satisfiable if and only if there exists a valid assignment in $\langle U, R, C, URQ \rangle$. On the one hand, assume that T is a truth assignment that satisfies ϕ. We now construct a valid assignment for $\langle U, R, C, URQ \rangle$. For every $i \in [1,n]$, if v_i is true, we assign $a_{i,1}$ to u_i and $a_{i,0}$ to u'; otherwise, if v_i is false, we assign $a_{i,0}$ to u_i and $a_{i,1}$ to u'. Also, if $a_{i,1}$ (respectively, $a_{i,0}$) is assigned to u_i, then b_j is assigned to u_i if and only if ϕ_j contains v_i (respectively, $\neg v_i$); that is to say, b_j is assigned to u_i if and only if setting v_i to true (respectively, false) satisfies the clause ϕ_j. Since every $\phi_j (j \in [1,m])$ is satisfied by T, every role $r_j \in R_b$ is assigned to at least one user. Also, every role in R_a is assigned to one user, and no mutual exclusion constraint is violated in our assignment. Therefore, the assignment is valid.

On the other hand, assume that there exists a valid assignment under $\langle U, R, C, URQ \rangle$. For every $i \in [1,n]$, u_i and u' are qualified for $a_{i,0}$ and $a_{i,1}$. Since $a_{i,0}$ and $a_{i,1}$ are mutually exclusive, one of them is assigned to u_i and the other is assigned to u'. We construct a true assignment by setting variable v_i to false if and only if $a_{i,0}$ is assigned to u_i; otherwise, we set v_i to true. Now, we prove that every clause in ϕ is satisfied by the truth assignment. Assume that b_i is assigned to u_j. Since u_j is qualified for b_i, according to our construction, either v_j or $\neg v_j$ appears in ϕ_i. Without loss of generality, assume that v_j appears in ϕ_i. In this case, according to our construction, b_i and $a_{j,0}$ are mutually exclusive. Hence, u_j must have been assigned to $a_{j,1}$, which indicates that v_j is set to true, and ϕ_i is satisfied. This indicates that every clause in ϕ is satisfied, and thus, ϕ is satisfied. □

Lemma 19. $\mathrm{AFP}\ \langle \mathrm{RC} + \mathrm{PRE} : \mathrm{conj} \rangle$ *is* **NP***-hard.*

Proof. We reduce the **NP**-complete SET COVERING problem to $\mathrm{AFP}\langle \mathrm{RC} + \mathrm{PRE} \rangle$. In the SET COVERING problem, we are given a set $S = \{e_1, \ldots, e_m\}$, a family of sets $F = \{S_1, \ldots, S_n\}$, where $S_i \subset S$, and an integer k. We are asked whether there exists k elements in F, whose union is equivalent to S.

Given a SET COVERING instance, we construct such a configuration $\langle U, R, C, URQ \rangle$: let $U = \{u_1, \ldots, u_n\}$ and $R = \{r', r_1, \ldots, r_m\}$. In URQ, every user in U is qualified for r'; user u_i is qualified for r_j if and only if $e_j \in S_i$. Intuitively, $u_i \in U$ corresponds to $S_i \in F$, and $r_j \in R$ corresponds to $e_j \in S$. For every $j \in [1,m]$, we specify a prerequisite constraint $\mathrm{PRE}(r', r_j)$ (i.e., a member of r_j must be a member of r'). Finally, we specify a role-cardinality constraint $\mathrm{RC}(r', 1, k)$.

Now, we prove that the answer to the SET COVERING instance is "yes" if and only if there exists a valid assignment under $\langle U, R, C, URQ \rangle$. On the one hand, without loss of generality, assume that the union of $S_1, \ldots, S_k$ is S. We now create a valid assignment for $\langle U, R, C, URQ \rangle$. We assign r' to $u_1, \ldots, u_k$, and for every $i \in [1,k]$, we assign to u_i all the roles she is qualified for. For every $i \in [1,m]$, since $e_i \in (S_1 \cup \cdots \cup S_k = S)$, according to our construction, r_i is assigned to at least one user in $\{u_1, \ldots, u_k\}$. Because r' is assigned to k users, constraint $\mathrm{RC}(r', 1, k)$ is satisfied. Also, since we assign roles only to $u_1, \ldots, u_k$ and all of them are assigned to r', no prerequisite constraint is violated. Therefore, the assignment is valid.

On the other hand, assume that there is a valid assignment under $\langle U, R, C, URQ \rangle$. According to the cardinality constraint, r' is assigned to at most k users. Without loss of generality, assume that r' is assigned to $u_1, \ldots, u_k$ in a valid assignment. Since the assignment is valid, for every $i \in [1,m]$, r_i is assigned to at least one user. Assume that r_i is assigned to u_j. According to the

prerequisite constraint $\mathrm{PRE}(r', r_i), u_j$ must be assigned to r' as well, which indicates that $j \in [1, k]$. In our construction, r_i corresponds to e_i and u_j corresponds to S_j. Therefore, for every $i \in [1, m], e_i \in (S_1 \cup \cdots \cup S_k)$. Hence, $S_1 \cup \cdots \cup S_k = S$ and the answer to the SET COVERING instance is "yes." □

ACKNOWLEDGMENTS

The first author's research was supported in part by the National High Technology Research and Development Program (863 Program) of China (2006AA01A113) and the Science Foundation of Shandong Province (Y2008G28). Portions of this work were also supported by the US National Science Foundation (NSF) grant 0712846 IPS: Security Services for Healthcare Applications, the MURI award FA9550-08-1-0265 from the Air Force Office of Scientific Research, the NSF Grants CNS-0915436, CNS-0913875, and Science and Technology Center CCF-0939370, by Grant FA9550-09-1-0223 from the Air Force Office of Scientific Research, and by sponsors of the Center for Education and Research in Information Assurance and Security (CERIAS). Part of first and second author's work was done while visiting Purdue University.

REFERENCES

[1] G.-J. Ahn and R.S. Sandhu, "The RSL99 Language for Role-Based Separation of Duty Constraints," *Proc. Fourth Workshop Role-Based Access Control,* pp. 43-54, 1999.

[2] G.-J. Ahn and R.S. Sandhu, "Role-Based Authorization Constraints Specification," *ACM Trans. Information and System Security,* vol. 3, no. 4, pp. 207-226, Nov. 2000.

[3] ANSI, *American National Standard for Information Technology—Role Based Access Control,* p. 359, ANSI Int'l Committee for Information Technology Standards, Feb. 2004.

[4] E. Bertino, E. Ferrari, and V. Atluri, "The Specification and Enforcement of Authorization Constraints in Workflow Management Systems," *ACM Trans. Information and System Security,* vol. 2, no. 1, pp. 65-104, Feb. 1999.

[5] H. Chen and N. Li, "Constraint Generation for Separation of Duty," *Proc. Ninth ACM Symp. Access Control Models and Technologies (SACMAT),* pp. 130-138, June 2006.

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms.* MIT Press, 2002.

[7] J. Crampton, "Specifying and Enforcing Constraints in Role-Based Access Control," *Proc. ACM Symp. Access Control Models and Technologies (SACMAT),* pp. 43-50, June 2003.

[8] J. Crampton, "A Reference Monitor for Workflow Systems with Constrained Task Execution," *Proc. ACM Symp. Access Control Models and Technologies (SACMAT),* pp. 38-47, June 2005.

[9] V.D. Gligor, S.I. Gavrila, and D.F. Ferraiolo, "On the Formal Definition of Separation-of-Duty Policies and Their Composition," *Proc. IEEE Symp. Research in Security and Privacy,* pp. 172-183, May 1998.

[10] IBM Tivoli Identity Manager 5.1. http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.itim.doc/cpt/cpt_ic_release_oview_whatsnew.html, 2009.

[11] T. Jaeger, "On the Increasing Importance of Constraints," *Proc. ACM Workshop Role-Based Access Control (RBAC),* pp. 33-42, 1999.

[12] T. Jaeger and J.E. Tidswell, "Practical Safety in Flexible Access Control Models," *ACM Trans. Information and System Security,* vol. 4, no. 2, pp. 158-190, May 2001.

[13] N. Li, M.V. Tripunitara, and Z. Bizri, "On Mutually Exclusive Roles and Separation of Duty," *ACM Trans. Information and System Security,* vol. 10, no. 2, May 2007.

[14] N. Li, M.V. Tripunitara, and Q. Wang, "Resiliency Policies in Access Control," *Proc. ACM Conf. Computer and Comm. Security (CCS),* Nov. 2006.

[15] D.L.B. (Project Leader) "Sat4j: A Satisfiability Library for Java," URL http://www.sat4j.org/, Jan. 2006.

[16] R.S. Sandhu, V. Bhamidipati, and Q. Munawer, "The ARBAC97 Model for Role-Based Aministration of Roles," *ACM Trans. Information and Systems Security,* vol. 2, no. 1, pp. 105-135, Feb. 1999.

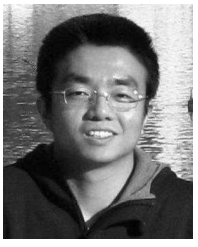
[17] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman, "Role-Based Access Control Models," *Computer,* vol. 29, no. 2, pp. 38-47, Feb. 1996.

[18] T.T. Simon and M.E. Zurko, "Separation of Duty in Role-Based Environments," *Proc. 10th Computer Security Foundations Workshop,* pp. 183-194, June 1997.

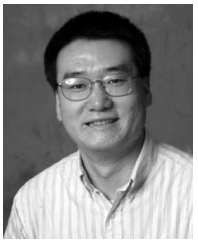
[19] J. Tidswell and T. Jaeger, "An Access Control Model for Simplifying Constraint Expression," *Proc. ACM Conf. Computer and Comm. Security,* pp. 154-163, 2000.

[20] Q. Wang and N. Li, "Satisfiability and Resiliency in Workflow Systems," *Proc. European Symp. Research in Computer Security (ESORICS),* Sept. 2007.

Yuqing Sun received the BSc, Master's, and PhD degrees in computer science from Shandong University, China. She is currently a professor in the School of Computer Science and Technology, Shandong University. She was a visiting scholar at Hong Kong University and at Purdue University. Her research interests include access control, security policy, privacy protection, social network, web services, and workflow management. She has published more than 30 papers in refereed journals and in international conferences and symposia proceedings. She has also served as a reviewer for international journals and served on the program committees of many international conferences.

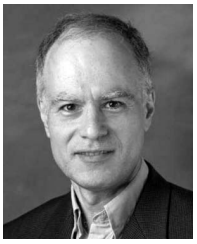


Qihua Wang received the BS degree in computer science from the University of Science and Technology of China, Hefei, in 2004, and the MS and PhD degrees in computer science from Purdue University, in 2007 and 2009, respectively. He joined the IBM Almaden Research Center in 2009. His research interests include information security, knowledge management, and social computing. He has published more than 20 technical papers in refereed journals and conference proceedings and served on the program committees of several international conferences. He was the recipient of the Diamond Award for Academic Excellence from the Center for Education and Research in Information Assurance and Security (CERIAS).



Ninghui Li received the BEng degree in computer science from the University of Science and Technology of China, Hefei, in 1993, and the MSc and PhD degrees in computer science from New York University, in 1998 and 2000, respectively. He is currently an associate professor in computer science at Purdue University. Prior to joining Purdue University in 2003, he was a research associate at the Computer Science Department, Stanford University. His research interests include security and privacy in information systems, with a focus on access control. He has worked on projects on trust management, automated trust negotiation, role-based access control, privacy-preserving data publishing, and operating system access control. He has published more than 90 technical papers in refereed journals and conference proceedings and has served on the program committees of more than four dozen international conferences and workshops. He is a senior member of the IEEE and a member of the ACM.

Elisa Bertino is currently a professor of computer science at Purdue University, and serves as research director of the Center for Education and Research in Information Assurance and Security (CERIAS). Previously she was a faculty member at the Department of Computer Science and Communication, University of Milan, where she was the department head and director of the DB&SEC laboratory. She has been a visiting researcher at the IBM Research Laboratory (now Almaden), San Jose, at Microelectronics and Computer Technology Corporation, at Rutgers University, and at Telcordia Technologies. Her research interests include security, privacy, digital identity management systems, database systems, distributed systems, and multimedia systems. In these areas, she has published more than 400 papers in all major refereed journals, and in proceedings of international conferences and symposia. She is a coauthor of the books *Object-Oriented Database Systems—Concepts and Architectures* (Addison-Wesley International Publ., 1993), *Indexing Techniques for Advanced Database Systems* (Kluwer Academic Publishers, 1997), *Intelligent Database Systems* (Addison-Wesley International Publ., 2001), and *Security for Web Services and Service Oriented Architectures* (Springer, 2009). She has been a coeditor-in-chief of the *Very Large Database Systems (VLDB) Journal* from 2001 to 2007. She serves, or has served, on the editorial boards of several scientific journals, including *IEEE Internet Computing*, *IEEE Security and Privacy*, the *IEEE Transactions on Knowledge and Data Engineering*, *ACM Transactions on Information and System Security*, *ACM Transactions on Web*, *Acta Informatics*, and the *Parallel and Distributed Database Journal*. She is a fellow of the IEEE and the ACM and has been been named a golden core member for her service to the IEEE Computer Society. She was the recipient of the 2002 IEEE Computer Society Technical Achievement Award for outstanding contributions to database systems and database security and advanced data management systems and the 2005 IEEE Computer Society Tsutomu Kanai Award for pioneering and innovative research contributions to secure distributed systems. She is currently serving on the Board of Governors for the IEEE Computer Science Society.



Mikhail (Mike) J. Atallah received the PhD degree from The Johns Hopkins University in 1982. He joined the Computer Sciences Department, Purdue University, where he currently holds the rank of distinguished professor of computer science. He is a fellow of the IEEE and the ACM. He has served on the editorial boards of top journals, and on the program committees of top conferences and workshops. He was keynote and invited speaker at many national and international meetings, and a speaker nine times in the Distinguished Colloquium Series of top Computer Science Departments. He was selected in 1999 as one of the best teachers in the history of Purdue University and included in *Purdue's Book of Great Teachers*, a permanent wall display of Purdue's best teachers past and present. He is a cofounder of Arxan Technologies Inc.