Naveen Jindal School of Management

2014-07

Role Refinement in Access Control: Model and Analysis

UTD AUTHOR(S): Milind W. Dawande and Vijay S. Mookerjee

©2014 INFORMS

Find more research and scholarship conducted by the Naveen Jindal School of Management here. This document has been made available for free and open access by the Eugene McDermott Library. Contact libwebhelp@utdallas.edu for further information.

INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Role Refinement in Access Control: Model and Analysis

Hao Xia, Milind Dawande, Vijay Mookerjee

To cite this article:
Hao Xia, Milind Dawande, Vijay Mookerjee (2014) Role Refinement in Access Control: Model and Analysis. INFORMS Journal on Computing 26(4):866-884. http://dx.doi.org/10.1287/ijoc.2014.0603

Full terms and conditions of use: http://pubsonline.informs.org/page/terms-and-conditions

This article may be used only for the purposes of research, teaching, and/or private study. Commercial use or systematic downloading (by robots or other automatic processes) is prohibited without explicit Publisher approval, unless otherwise noted. For more information, contact permissions@informs.org.

The Publisher does not warrant or guarantee the article's accuracy, completeness, merchantability, fitness for a particular purpose, or non-infringement. Descriptions of, or references to, products or publications, or inclusion of an advertisement in this article, neither constitutes nor implies a guarantee, endorsement, or support of claims made of that product, publication, or service.

Copyright © 2014, INFORMS

Please scroll down for article—it is on subsequent pages

INFORMS is the largest professional society in the world for professionals in the fields of operations research, management science, and analytics.
For more information on INFORMS, its publications, membership, or meetings visit http://www.informs.org

INFORMS Journal on Computing
Vol. 26, No. 4, Fall 2014, pp. 866–884
ISSN 1091-9856 (print) | ISSN 1526-5528 (online)

http://dx.doi.org/10.1287/ijoc.2014.0603
© 2014 INFORMS

Role Refinement in Access Control: Model and Analysis

Hao Xia
School of Management, Harbin Institute of Technology, Harbin, PR China, 150001, xiah@utdallas.edu

Milind Dawande, Vijay Mookerjee
Naveen Jindal School of Management, The University of Texas at Dallas, Richardson, Texas 75080
{milind@utdallas.edu, vijaym@utdallas.edu}

Access control mechanisms in software systems administer user privileges by granting users permission to perform certain operations while denying unauthorized access to others. Such mechanisms are essential to ensure that important business functions in an organization are conducted securely and smoothly. Currently, the dominant access control approach in most major software systems is role-based access control. In this approach, permissions are first assigned to roles, and users acquire permissions by becoming members of certain roles. However, given the dynamic nature of organizations, a fixed set of roles usually cannot meet the demands that users (existing or new) have to conduct business.

The typical response to this problem is to myopically create new roles to meet immediate demand that cannot be satisfied by an existing set of roles. This ad hoc creation of roles invariably leads to a proliferation in the number of roles with the accompanying administrative overhead. Based on discussions with practitioners, we propose a role refinement scheme that reconstructs a system of roles to reduce the cost of role management. We first show that the role-refinement problem is strongly NP-hard and then provide two polynomial-time approximation algorithms (a greedy algorithm and a randomized rounding algorithm) and establish their performance guarantees. Finally, numerical experiments—based on a real data set from a firm's enterprise resource planning system—are conducted to demonstrate the applicability and performance of our refinement scheme.

Keywords: software systems; role-based access control; role refinement; approximation algorithms
History: Accepted by Alexander Tuzhilin, (former) Area Editor for Knowledge and Data Management; received May 2013; revised October 2013, January 2014; accepted February 2014. Published online in *Articles in Advance* July 28, 2014.

1. Introduction

In most modern enterprises, business processes have become deeply integrated into software systems (Bai et al. 2011). Examples of such systems include enterprise resource planning (ERP) systems, which are information systems that integrate the various departments and functions and support the internal business processes of organizations. To accomplish business processes, end users must possess certain specified privileges and capabilities within software systems, corresponding to their current identity in the organization. Thus, access control is essential to ensure the security and effective operation of a software system and the business processes it supports (Bai et al. 2012). Access control mechanisms in software systems administer the privileges of different users by granting them permissions to perform specific operations and denying unauthorized access to others (Li and Wang 2008). As business processes become increasingly integrated and complex, a critical challenge for software systems is to implement a properly designed access control mechanism.

The need for access control in software systems is further emphasized through regulations enforced by the government, such as the Sarbanes-Oxley Act (SOX). SOX, a U.S. federal law enacted in 2002, arose in response to major corporate and accounting scandals at Enron, Tyco International, WorldCom, etc. This act established enhanced standards for all U.S. public company boards, management, and public accounting firms. A cornerstone of SOX is *separation of duties* (SoD), a principle that disseminates the tasks and associated privileges of a specific business process among multiple users to prevent fraud and errors (Botha and Eloff 2001). This principle, deemed to be the one of the most fundamental principles of SOX (Coleman 2008), requires that a company and its external auditor(s) report on the adequacy of the company's internal control over financial reporting. In businesswide software systems (including, but not necessarily limited to ERP systems), the access control mechanism is mandated by SOX to implement the SoD principle.

There are different approaches to access control, including mandatory access control, discretionary access control, and role-based access control (RBAC).

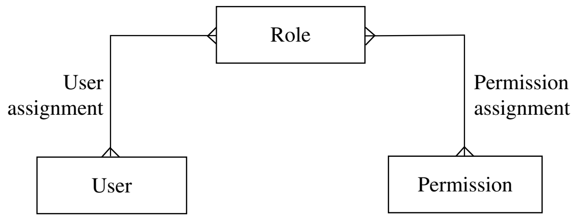


Figure 1 The Basic RBAC Model

The relatively new RBAC model, formalized by Ferraiolo and Kuhn in 1992, is the most flexible and powerful access control framework and can simulate mandatory access control (Osborn et al. 2000) and discretionary access control (Sandhu and Munawer 1998). In addition to its flexibility, another advantage of RBAC is that it can efficiently implement the SoD principle (Ferraiolo et al. 2001). As a result, RBAC has become the dominant practice for advanced access control and is implemented by most major software vendors, including SAP, IBM, and Oracle (Colantonio et al. 2011). This paper is based on the RBAC model (implemented in SAP R/3), which we discuss next.

1.1. Role-Based Access Control

In the RBAC model, there are three basic entities: *permissions*, *role*, and *user*. Permissions are assigned to roles; i.e., each role is a collection of permissions; each active user is assigned to certain roles and therefore acquires the union of permissions from these roles. The permission-role assignment and the user-role assignment are both many-to-many relations, granting RBAC high flexibility. Advanced features such as role hierarchy are made available in more sophisticated implementations, but the basic model shown in Figure 1 suffices for our current discussion.

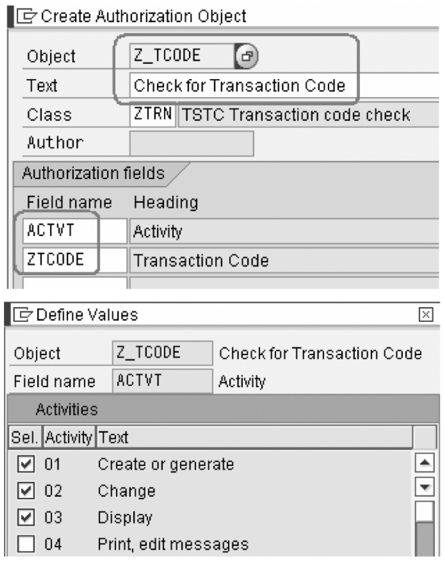
Roles are at the core of the RBAC model and work as an intermediary between users and permissions. Roles are a convenient way to implement data abstraction, similar to the concept of an interface in object-oriented systems (Sandhu et al. 1996, 2000). Data abstraction is the principle of representing information in a manner that hides unnecessary details (usually as it pertains to implementation) yet provides enough information for humans to understand its semantics. A particular permission in an ERP system can be quite fine-grained, involving technical and implementation details (e.g., the right to view or change a particular field in a specific table of a database). To make matters worse, it may be difficult for a user-manager (attempting to assign permissions to end users) to anticipate the combined security impact of a set of permissions. Instead, it is much easier for administrators to think and reason in terms of a collection of permissions (or roles) rather than to make assignments at the level of individual permissions. A role is a meaningful collection of permissions that is required to perform a certain task. For instance, a role (such as "manage inventory at a specific warehouse") may represent a collection of permissions required to check inventory information and reorder (if necessary) certain products in that warehouse. Simply put, roles are small but meaningful chunks of permissions, where the term "meaningful" refers to a cohesive function or activity.

The number of users and permissions in a large enterprise are usually in the thousands (or even more), but the number of roles is much smaller (Ferraiolo et al. 2007). If a user's job responsibilities can be decomposed into smaller functional units that correspond to existing roles, then a system administrator can simply assign these roles to the user instead of dealing with (possibly hundreds of) specific permissions.

It is common for a role to possess more than one permission and to have overlapping permissions with other roles. Similarly, users often play more than one role and a role can be performed by multiple users. Under RBAC, the permission-role assignment (referred to as the role system, consisting of a set of roles) is expected to be relatively static, but the user-role assignment is usually updated on demand by system administrators. This approach is both efficient and secure. A National Institute of Software Standards and Technology (NIST) study indicates that it is desirable not to allow the user-role assignment administrators to modify permission-role assignment (Ferraiolo et al. 1993).

An RBAC system must be able to exert specific constraints on both permission-role and user-role assignments to conform to an organization's policies and implement mandatory regulations (Basu and Kumar 2002). In addition, the RBAC system must be designed to implement SoD mandates. To this end, the first requirement is that any given role in the role definition be SoD compliant. Thus, it should not be a violation of SoD to allow any user to perform the permissions associated with any single role. The SoD principle would also need to constrain user-role assignments, namely, prohibit a single user from being assigned a pair of conflicting roles. Another, more general constraint, referred to collectively as statically mutually exclusive roles, requires that no user be allowed to be a member of more than a specified number of roles in a given collection of roles (Li et al. 2007).

To illustrate how RBAC works, Figure 2 presents screenshots taken from SAP's Netweaver (Release 7). The first screenshot shows a newly created authorization object that authorizes a user to carry out a certain action. An authorization object is equivalent to the concept of a permission in the RBAC model. In the second screenshot, a new role is created by assigning permissions (authorization objects) to the role. The last screenshot shows how roles can be assigned to users. The permissions a user holds are, of course, easily determined given her membership in certain



Create Authorization Object

Object	Z_TCODE
Text	Check for Transaction Code
Class	ZTRN TSTC Transaction code check
Author	

Authorization fields

Field name	Heading
ACTVT	Activity
ZTCODE	Transaction Code

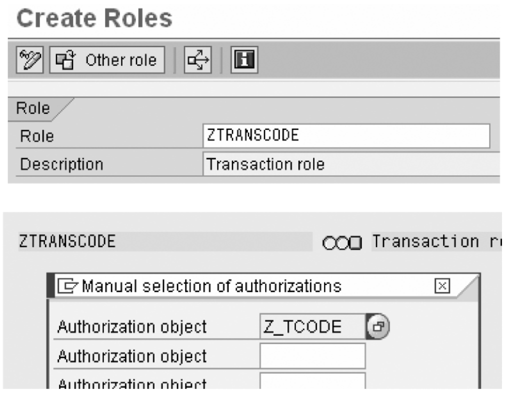
Define Values

Object	Z_TCODE	Check for Transaction Code
Field name	ACTVT	Activity

Activities

Sel.	Activity	Text
☑	01	Create or generate
☑	02	Change
☑	03	Display
☐	04	Print, edit messages

Create authorization object



Create Roles

Role

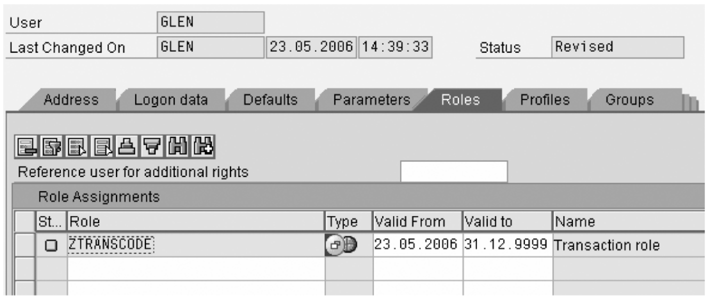
Role	ZTRANSCODE
Description	Transaction role

ZTRANSCODE Transaction r

Manual selection of authorizations

Authorization object	Z_TCODE
Authorization object	
Authorization object	

Create roles



User	GLEN				
Last Changed On	GLEN	23.05.2006	14:39:33	Status	Revised

Reference user for additional rights

Role Assignments

St...	Role	Type	Valid From	Valid to	Name
	ZTRANSCODE		23.05.2006	31.12.9999	Transaction role

Assign roles to user

Figure 2 Role-Based Access Control in SAP Netweaver 2004s, Release 7

roles, and the permissions corresponding to these roles. Any transaction in the system is associated with an identification code (T-code) and the permissions it requires. When a user tries to perform this transaction, an authorization check takes place by comparing the permissions the user holds and the permissions required by the transaction.

1.2. Role Refinement: Motivation and Objective

The user-role mapping in a RBAC system needs to be dynamically updated to cater to the changing work arrangements of the personnel in an organization. Consider a user who requires a certain set of permissions to accomplish a new or modified business process. These permissions must be exactly covered by a set of roles such that, if the user is assigned these roles, the necessary permissions—no more, no less—would be obtained. Holding more permissions than necessary is against the principle of least privilege and can lead to a potential security concern (Saltzer 1974, Ferraiolo and Kuhn 1992, Schneider 2000, Takabi et al. 2010). On the other hand, insufficient permissions could obstruct the execution of a business process. A question of feasibility naturally arises: given the current role system (i.e., the existing set of roles), does such a subset of roles already exist? Unfortunately, the answer to this question is often negative. Therefore, the role system needs frequent updates to ensure feasibility. Because it is not easy to anticipate future requirements when adding new roles to meet current needs, this process is myopic: when feasibility issues occur, new roles are created to meet the need. As a result, the role system in a RBAC implementation often grows rapidly over time (O'Connor and Loomis 2010, Elliott and Knight 2010).

This situation motivates the problem in this study. Although storage cost and the cost of associated computing resources constitute a relatively insignificant component of the total cost incurred in managing a role system, the dominant component is the cost incurred in the effort required to manage roles. Role managers need to simultaneously administer the roles in a RBAC system and verify the requests of permissions by users. Note that managing the role model is a more specialized and skilled task (and hence more expensive) than the direct allocation of access privileges to users. As the size of the role system grows, the cost of the managerial effort can increase significantly. As an unwelcome side effect, other business problems arise from role management staff that is overburdened with role administration tasks. These problems include (i) slow activation for new users, (ii) privilege accumulation as users move through an organization, and (iii) slow and unreliable access termination. Hence, a desirable objective is to minimize the number of roles in the system (Phillips 2009, Frank et al. 2010). Of course, such a measure of system cost implies that each role incurs the same cost. A more sophisticated measure of cost is to assign a specific cost to each role, determined, for example, by the number of permissions in the role. The total cost a role system incurs is then the sum of the costs of all its roles.

Our communications with practitioners indicate that system administrators need to periodically perform the task of role refinement (Saviynt 2012). This task creates a new role system that performs all the functions of the existing one but does so at a lower cost. To better understand this task, consider a simple instance of a RBAC system with only five unique permissions represented by the serial numbers 1–5. Assume that the current role system has four roles—$\{1,2\}$, $\{3,4\}$, $\{5\}$, and $\{1,2,4,5\}$—and that there are five users whose demands for permissions are $\{1,2,3,4\}$, $\{1,2,3,4,5\}$, $\{3,4\}$, $\{1,2,4,5\}$, and $\{3,4,5\}$, respectively. Although the demands of the users can be satisfied by the current role definition, it is easy to see that a new role definition with only three roles—$\{1,2\}$, $\{3,4\}$, and $\{4,5\}$—is also feasible. This new role definition may be preferable because of its smaller size.

The instance of the role refinement problem presented above may seem trivial, but a real RBAC system (a real data set is discussed in §4) may need to administer tens of thousands of permissions with thousands of roles and users. In addition, the number of permissions needed for a user could range from one to more than a thousand. Thus, in most practical cases, finding an optimal role system is not easy. Spurred by this need, and given the widespread use of RBAC in most modern software systems, our goal in this study is to find a practical solution to the role refinement problem. To the best of our knowledge, this problem has lacked formal examination and hence, motivates the current study.

1.3. Summary of Results and Novelty of Algorithmic Contribution

The role refinement problem (RRP), defined in §2, is the task of selecting roles from all eligible candidates to form a new role system that can meet the demand for permissions of all users, at minimum cost. To this end, each role in the existing role system must be expressible as an exact union of a subset of roles in the new role system. A theoretically equivalent variant of RRP, which we refer to as RRP^{U}, requires that only the permissions of each user (and not necessarily each role in the current role system) be expressible as an exact union of a subset of roles in the new role system. RRP (equivalently, RRP^{U}) is shown to be a generalization of the classical set-cover problem by allowing multiple nondisjoint target sets (roles) that need to be exactly covered. We show that both problems are strongly NP-hard and are also hard to approximate efficiently. In particular, unless $\mathbf{P} = \mathbf{NP}$, there does not exist a polynomial-time approximation algorithm than can achieve a guarantee better than $O(\ln M)$, where M is the sum of the cardinalities of all the roles in the existing role system.

In §3, we design two $O(\ln M)$-factor, polynomial-time approximation algorithms to solve RRP and RRP^{U}, as follows.

1. Greedy: This algorithm iteratively puts a candidate role with the lowest cost per unit coverage into a set of roles, until this set becomes a feasible solution. The cost of this solution is shown to be $O(\ln M)$ times the cost of the optimal solution.
2. Randomized Rounding: This algorithm obtains a solution by combining $\lceil 2\ln M \rceil$ potential solutions by randomly rounding the optimal solution of the linear programming (LP) relaxation corresponding to a specific integer programming (IP) formulation. We show that this solution is feasible with high probability for sufficiently large M and its expected cost is $O(\ln M)$ times the cost of the optimal solution.

In §4 we report on our experience with these algorithms for both RRP and RRP^{U} on a test bed that uses real data from a firm's existing RBAC system. Our primary conclusion is that both algorithms provide refined role systems with significantly lower costs than the existing role system does and they do so in a reasonable amount of computing time.

Novelty of Algorithmic Contribution: RRP is a proper generalization of the classical set-cover problem. First, there are multiple nondisjoint sets (usually in the hundreds) to be covered simultaneously. Second, each set should be covered exactly by some collection of chosen subsets of the base set (i.e., each set should be exactly the union of some collection of chosen subsets of the base set). The former condition is nonexistent for the set-cover problem. The latter condition is trivially satisfied for the set-cover problem, since there is only one set to be covered. Although the greedy algorithm and the randomized algorithm are well known for the set-cover problem, a surprising result of our analysis is that, with some appropriate modifications, these two algorithms work for problem RRP, which is significantly more general than the set-cover problem.

1.4. Review of Relevant Literature

The first formal RBAC model and its technical specifications were introduced by Ferraiolo and Kuhn (1992) at the NIST. The RBAC model was then expanded to include features such as role hierarchies and constraints (e.g., cardinality constraints, mutually exclusive roles, etc.) by Ferraiolo et al. (1995) and Sandhu et al. (1996), who integrated their work and proposed a unified NIST standard model for RBAC in 2000 and a revised version in 2001. A formal American National Standard for RBAC (ANSI/INCITS 359-2004) was approved in 2004.

From a technical perspective, there are several studies that formally analyze RBAC to verify the consistency of security policies that the model can implement (Shafiq et al. 2005) and to generate constraints on mutually-exclusive roles that enforce the principle of separation of duty. A considerable body of literature exists on designing new RBAC systems to reduce administrative complexity and cost and to support security policies and regulations (e.g., Ferraiolo et al. 1999, Botha and Eloff 2001, O'Connor and Loomis 2010). The design of a set of roles is a major step in the implementation of a RBAC system and is commonly referred to as "role engineering"; the objective of this is to ensure that the user-permission assignment stays unchanged after the organization switches to RBAC; see, e.g., Coyne (1995) and O'Connor and Loomis (2010). The use of data mining techniques for role engineering was proposed by Kuhlmann et al. (2003). In subsequent studies, the term "role mining" has been associated with automated approaches to role engineering (Frank et al. 2010).

The role mining problem (RMP) and its variants and extensions has been an active domain of research over the past several years. A formal description of RMP and several of its variants is provided by Vaidya et al. (2010), where it is broadly defined as the problem of "discovering an optimal set of roles from existing user permissions" (p. 1). More formally, the inputs to the RMP include the set of users, the set of permissions, and the many-to-many mapping of user-permission assignments (UPA). The output of RMP is a RBAC configuration that consists of a set of roles (ROLES), the many-to-many mapping of role-permission assignments (PA), and the many-to-many mapping of user-role assignments (UA).

The primary constraint of RMP is that PAs and UAs should, together, grant the same user-permission assignments as UPA, i.e., $\text{UA} \times \text{PA} = \text{UPA}$, and the objective is to minimize the total number of roles, i.e., $|\text{ROLES}|$. Lu et al. (2008) and Ene et al. (2008) extend this objective to include the number of user-role assignments (i.e., $|\text{UA}|$) and the number of role-permission assignments (i.e., $|\text{PA}|$). Role hierarchy (RH), as part of the RBAC configuration, has also been investigated in the RMP literature. RH is a partial order over the set of roles, where the senior role inherits the permissions of its junior roles. Finding a desirable RH that contains the minimal number of such partial orders is deemed as (a part of) the objective of RMP (Guo et al. 2008, Takabi and Joshi 2010, Colantonio et al. 2010). Molloy et al. (2008) propose a "weighted structural complexity (WSC)" as a general measure of goodness of the result of role mining. WSC is the weighted sum of $|\text{ROLES}|$, $|\text{UA}|$, $|\text{PA}|$, $|\text{UPA}|$, and $|\text{RH}|$. By adjusting the weights of these different components, WSC can simulate a variety of optimization objectives. Colantonio et al. (2008, 2009b) consider cost functions that simultaneously measure the system administration effort as well as the business meaning of the roles. Colantonio et al. (2008) consider a cost function that is a weighted sum of $|\text{ROLES}|$, $|\text{UA}|$, $|\text{PA}|$, and $\sum_{r\in\text{ROLES}} g(r)$, where the function g: $\text{ROLES} \to \mathbb{R}$ captures the additional cost from business information that is different from $|\text{ROLES}|$, $|\text{UA}|$, and $|\text{PA}|$ (e.g., separation of duties rules, incompatible combinations of permissions, whether a role is used exclusively in one organizational unit or across multiple organizational units). Colantonio et al. (2009b) define two metrics to evaluate the business meaning of roles: *activity spread* and *organization unit spread*. The first metric is based on the idea that the business meaning of a role increases if its activities are tightly related to each other, and the second captures the idea that the business meaning of a role decreases as the number of organizational units that use the role increases. The authors then consider a cost function that is a weighted sum of $|\text{ROLES}|$, $|\text{UA}|$, the total activity spread, and the total organization unit spread of the role system.

Several RMP variants relax the abovementioned primary constraint and allow for a limited amount of difference between $\text{UA} \times \text{PA}$ and UPA or set the objective to be one of minimizing this difference under the constraint that the number of roles should be bounded from above by a given number; see, e.g., Lu et al. (2008), Vaidya et al. (2010), and Frank et al. (2013). Lu et al. (2008) formulate the basic RMP and several of its variants using binary integer programming and provide heuristic solutions obtained via greedy algorithms.

Vaidya et al. (2008), Guo et al. (2008), and Takabi and Joshi (2010) extend the RMP by discussing the scenario where a deployed role system already exists and the "cost of migrating" from the deployed role system to a new role system needs to be considered. The cost of migration is assumed to decrease with an increase in the similarity between the deployed role system and the new role system. These papers propose several different approaches for calculating a dissimilarity score or distance (D) between two role systems and use it to define the objective of role mining: (1) Guo et al. (2008) study the minimal perturbation role hierarchy problem that minimizes the weighted sum of $|\text{RH}|$ and D; (2) Vaidya et al. (2008) analyze the minimal perturbation RMP that minimizes the weighted sum of $|\text{ROLES}|$ and D; (3) Takabi and Joshi (2010) study the mining role hierarchy with minimal perturbation problem that minimizes the weighted sum of WSC and D.

Colantonio et al. (2009a) investigate the basic RMP and, by leveraging its equivalence to the vertex coloring problem, prove that the optimal result (minimum number of roles) is concentrated around its expectation. This can help determine a suitable stopping condition when employing probabilistic algorithms for the basic RMP. Frank et al. (2008, 2013) propose a probabilistic scheme to solve the role mining problem. Different from most RMP studies that try to compress the UPAs into UAs and PAs, they propose that RMP is more an inference problem than a data compression problem. More specifically, their scheme assumes that there exists an underlying, yet hidden RBAC configuration that is influenced by managerial information and induces the known UPA. Accordingly, the objective is to infer the RBAC configuration that most likely explains the known data: the current UPA and, optionally, a part of the managerial information. Their scheme therefore promotes hybrid role mining, which tries to utilize managerial information gained from a top-down analysis.

Our model and methods for RRP differ from existing RMP-related studies in the following three aspects:

1. The input to the RRP includes a candidate set of roles. New roles can only be selected from this candidate set of roles. This is an important requirement, since the goal of the RRP is to help firms that have existing RBAC systems improve the management of these systems. Thus, the candidate set of roles is, in general, heavily influenced by the current set of roles in use and the specific improvements being sought. The RMP, in general, does not have such a restriction and, theoretically, new roles can be constructed arbitrarily as sets of permissions. Although some studies on RMP propose creating a candidate set of roles by, for instance, generating pairwise intersections of the permissions of the users (Vaidya et al. 2006, 2010), it is a compromise to reduce solution complexity rather than a genuine model setting. Essentially, RRP is a deductive process, whereas RMP is an inductive process. Therefore, the underlying mathematical structures of RRP and RMP are quite different. On one hand, studies on RMP establish its equivalence to some well-known NP-hard combinatorial optimization problems, such as the minimum tiling problem (Vaidya et al. 2007) and the vertex coloring problem (Colantonio et al. 2009a), and existing solutions to these problems are borrowed to help develop solutions to RMP. On the other hand, RRP is a novel generalization of the classical set-cover problem.

2. The optimization objective of RRP is to minimize the total cost of the new role system, which is the sum of the costs of all the roles in the system. The cost of each candidate role can be any arbitrary nonnegative value and can be assigned by the RBAC system administrators according to various business needs and technical concerns. This naturally grants high flexibility to the model. The cost-minimization objective is different from the objectives in the existing literature; for instance, the general WSC measure discussed earlier cannot simulate this objective by adjusting weights.

3. We develop two polynomial-time approximation algorithms—a greedy algorithm and a randomized rounding algorithm—to solve RRP and prove that they provide the best possible performance guarantee for RRP in polynomial time, assuming $\mathbf{P} \neq \mathbf{NP}$. Although solutions based on greedy algorithms are common in solving RMP variants, no mathematical guarantees are available except for the basic RMP that has the objective of minimizing |ROLES| (Vaidya et al. 2010). Also, to our knowledge, the application of the randomized rounding technique is novel in this area.

The role-refinement problem in our study is a generalization of the classical set-cover problem. In RBAC, the core idea is that a user gains the necessary permissions through roles. A role is a set of permissions, and a user, for the purposes of access control, can also be viewed as a set of permissions. The set-cover problem is as follows: given $\mathcal{U}$, a universe of n elements and $\mathcal{F}$, and a collection of subsets of $\mathcal{U}$, find the lowest cost collection of subsets from $\mathcal{F}$ such that the union of these chosen subsets is $\mathcal{U}$. The decision problem corresponding to this optimization problem was one of the first problems proven to be NP-complete by Karp (1972). Several generalizations of the set-cover problem have also been investigated; see, e.g., Hall and Hochbaum (1986), Hall (1992), Vazirani (2001), Yang and Leung (2005), Har-Peled and Lee (2012), and Chekuri et al. (2009). To our knowledge, the generalization corresponding to RRP—in which multiple roles (usually in the thousands) each need to be covered exactly by a collection of nondisjoint subsets of permissions—is a novel contribution to the literature.

2. An Analytical Model of the Role Refinement Problem

We start by defining the RRP. A useful mathematically equivalent variant of RRP is then discussed in §2.2.

2.1. The Role-Refinement Problem

The purpose of the RRP is to create a new, refined system of roles to replace the existing one. We now describe the basic intuition and then the precise mathematical definition of the problem.

Consider an existing RBAC system. By definition, each role in the current system is a set of permissions. Let $\mathcal{P}$ denote the set of all permissions. Suppose that there are k roles in the current role system, denoted by $R_1, R_2, \ldots, R_k$. Thus, the existing role system is a set system $\mathcal{R} = \{R_1, R_2, \ldots, R_k\}$ with $R_i \subseteq \mathcal{P}, i = 1, 2, \ldots, k$. The output of the RRP is another set system $\mathcal{S} = \{S_1, S_2, \ldots, S_l\}$, with $S_i \subseteq \mathcal{P}$, $i = 1, 2, \ldots, l$. We want to design the new system $\mathcal{S}$ so that each role in the existing system $\mathcal{R}$ is exactly the union of some roles in $\mathcal{S}$. That is, $R_i = \bigcup_{j\in\gamma_i} S_j$ and $\gamma_i \subseteq \{1, 2, \ldots, l\}$. This is to ensure that the new system $\mathcal{S}$ has at least the same functionality as that offered by $\mathcal{R}$. The core function of a role system is to grant each user precisely the set of permissions she requires. Consider, for example, a user U to whom a set of roles, say $R_1, R_2, \ldots, R_p$, is assigned. Thus, the set of permissions for this user is $\bigcup_{i=1}^{p} R_i$. Since each role in $\mathcal{R}$ is exactly the union of some roles in $\mathcal{S}$, we have $\bigcup_{i=1}^{p} R_i = \bigcup_{i=1}^{p} \bigcup_{j\in\gamma_i} S_j$. Consequently, the new system $\mathcal{S}$ also has the capability to precisely provide to each user the same set of permissions the system $\mathcal{R}$ provided. In other words, there is no loss of functionality—as far as user permissions are concerned—in moving from $\mathcal{R}$ to $\mathcal{S}$.

The objective of the role refinement is to make the new system $\mathcal{S}$ as "compact" as possible. As mentioned in §1, firms incur costs in managing roles. Thus, a general objective is to obtain a least-cost role system $\mathcal{S}$. A special case of this objective, when the management of each role incurs the same cost, is to obtain a system $\mathcal{S}$ with the minimum number of distinct roles.

There are other important requirements of role refinement that restrict the choices of roles for the new system $\mathcal{S}$. First, the new RBAC system (after role refinement) should comply with the regulations that were respected in the existing RBAC system $\mathcal{R}$. For instance, if SoD compliance is critical, then all the roles in $\mathcal{S}$ should be SoD compliant. Second, as mentioned in §1, a role is often designed to serve a specific business purpose and should, therefore, be transparent and easily understood across the firm. Third, the RBAC software the firm adopts may impose some operational constraints, e.g., an upper limit on the number of permissions a role can contain. All these requirements naturally restrict the available choices for new roles. We assume that the set of all eligible candidate roles—which have passed these and any other applicable requirements—is an input to the RRP. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ be the set of all candidate roles. Also, we denote the cost of managing role $C_i \in \mathcal{C}$ by $c(C_i)$. Approaches for generating $\mathcal{C}$ are discussed in §2.3.

We are now ready to formally define the RRP.

Role-Refinement Problem

Instance: A triplet $\{\mathcal{R}, \mathcal{C}, c\}$, where (i) $\mathcal{R} = \{R_1, R_2, \ldots, R_k\}$ is the existing role system, (ii) $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ is the candidate set of roles available for the new (desired) system, and (iii) $c: \mathcal{C} \to \Re_+$ is the cost function of the roles.

Solution: A new role system $\mathcal{S} \subseteq \mathcal{C}$, such that *each* role in $\mathcal{R}$ is exactly the union of some subset of roles in $\mathcal{S}$.

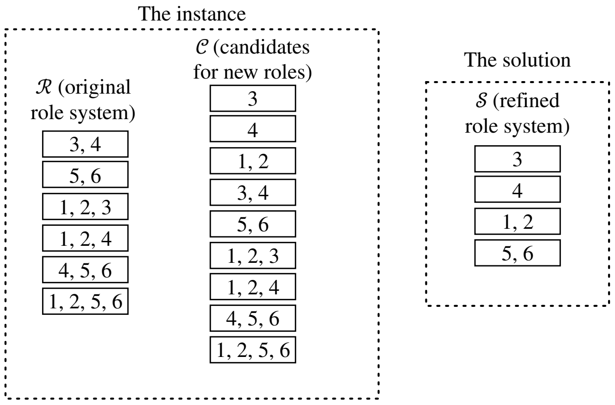


Figure 3 **An Instance of Role Refinement Problem and Its Solution**
Note. For simplicity, the cost function is to minimize the number of roles in the refined role system (i.e., $c(\cdot) = 1$).

Objective Function: Minimize the cost of $\mathcal{S}$, defined as the sum of the costs of the roles in $\mathcal{S}$. That is, minimize $\sum_{C_i\in\mathcal{S}} c(C_i)$.

Figure 3 illustrates a simple instance of the RRP and its solution, where the cost function $c(\cdot) = 1$. In this case, the objective becomes that of minimizing the cardinality of the new system $\mathcal{S}$.

As mentioned earlier, RRP is a generalization of the set-cover problem and is, therefore, strongly NP-hard. Specifically, if the existing role system $\mathcal{R}$ consists of a single role (i.e., $|\mathcal{R}| = 1$), then RRP reduces to the set-cover problem. We will exploit this connection in §3 to develop efficient approximation algorithms for RRP.

2.2. A Variant of RRP Based on Users

As previously defined, RRP requires that each role in the existing system $\mathcal{R}$ be expressible as an exact union of a subset of roles in the new system $\mathcal{S}$. This requirement ensures that $\mathcal{S}$ can precisely "cover" any set of permissions that $\mathcal{R}$ can. If we restrict attention to only the permissions of the current users, then we can instead define the following alternative problem, which is mathematically equivalent to RRP.

Let l be the number of active users in the RBAC system. Let U_i denote the set of permissions of user i, $i = 1, 2, \ldots, l$. We want to ensure that the set of permissions of each of these active users is precisely expressible in the new system $\mathcal{S}$. Thus, it is enough to require that each permission set U_i, $i = 1, 2, \ldots, l$, is exactly the union of some subset of new roles in system $\mathcal{S}$. The other requirements are the same as those for RRP. We refer to this user-based variant as RRP$^{\mathrm{U}}$ and define it formally, as follows.

Role Refinement Problem Based on Users (RRP$^{\mathrm{U}}$)

Instance: A triplet $\{\mathcal{U}, \mathcal{C}, c\}$, where (i) $\mathcal{U} = \{U_1, U_2, \ldots, U_l\}$ is the set system of permissions of the active users, (ii) $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ is the candidate set of roles available for the new (desired) system, and (iii) $c: \mathcal{C} \to \Re_+$ is the cost function of the roles.

Solution: A new role system $\mathcal{S} \subseteq \mathcal{C}$, such that *each* permission set in $\mathcal{U}$ is exactly the union of some subset of roles in $\mathcal{S}$.

Objective Function: Minimize the cost of $\mathcal{S}$, defined as the sum of the costs of the roles in $\mathcal{S}$. That is, minimize $\sum_{C_i \in \mathcal{S}} c(C_i)$.

It is clear that the two variants RRP and RRP^U are theoretically equivalent, as their underlying mathematical structure is the same. However, there are some differences in their practical usage. We mention two as follows:

1. In a rapidly changing business environment, the permissions assigned to current active users reflect the real demand of access control across the firm. The variant RRP^U focuses only on this real demand by ensuring that the new role system $\mathcal{S}$ is able to assign to each user her current set of permissions. In contrast, RRP does more: it ensures that each role in the existing role system, whether in active use, can be replicated by the new set of roles.

2. RRP may have a comparative advantage, in terms of the continuation of implementation of security measures, over RRP^U that derives from the fact that it maintains the structure of the existing role system after role refinement. For instance, existing user-role assignment constraints such as the statically mutually exclusive roles restrictions mentioned in §1 and prerequisite roles (Sandhu et al. 1996) can be more easily maintained following role refinement using RRP. Another practical consideration is the assignment of role managers for the new role system. A typical property of the existing role definitions is that each role manager's authority of supervision is below a certain security threshold. Following RRP, each new role is a subset of at least one existing role. This ensures that the same security thresholds are feasible for the new role definitions as well.

2.3. The Generation of Candidate Roles

As discussed, the set of candidate roles $\mathcal{C}$ is an input to RRP and RRP^U. We now discuss recent progress of role engineering techniques that can help generate $\mathcal{C}$.

There are two basic strategies of role engineering: *bottom up* and *top down*. In the bottom-up approach, roles are formulated based on the existing UPAs, as discussed in §1.4. Because bottom-up role engineering usually relies on automated techniques (e.g., data mining), it is often referred to as role mining. In the top-down approach, business processes in the enterprise are carefully analyzed and decomposed into meaningful functional units. Then the set of permissions required to perform a functional unit is defined as one role. Both approaches are inherently imperfect: on one hand, the top-down approach generates roles that are meaningful and easy to understand from a managerial perspective. However, (manually) analyzing all the functional units in a large enterprise can be time consuming and costly, making it infeasible in most cases as the only way to generate candidate roles. On the other hand, although many efficient role mining techniques have been proposed, the roles discovered by automated data mining methods may not always be meaningful (Molloy et al. 2008, Colantonio et al. 2009b, Vaidya et al. 2010).

To overcome this dilemma, researchers have developed hybrid approaches that try to improve role mining by utilizing managerial information gained from top-down analysis. To name a few, Molloy et al. (2008) propose a role mining method that discovers roles that can be associated with a user's attributes (e.g., job responsibilities, work locations, etc.) and thus are semantically sound. Another role mining scheme that exploits data on users' attributes to generate meaningful roles has been proposed by Xu and Stoller (2012). Colantonio et al. (2009b) develop a technique to identify meaningful roles that relies on the belief that a role is more likely to be meaningful when it supports activities within the same business process or is assigned to users in the same organizational unit. They propose schemes that first decompose the UPA information into partitions corresponding to business categories and then perform role mining on each partition, so that the elicited roles are more closely related with underlying business functions (Colantonio et al. 2011, 2012).

Commercial services for designing meaningful roles are also available. A good example is the product Enterprise Role Management from Saviynt (http://www.saviynt.com), a provider of access control solutions.

To end this section, we note that the feasibility of both RRP and RRP^U should not be an issue if we include all the existing roles in $\mathcal{R}$ in the set of candidate roles $\mathcal{C}$. In this case, the existing set of roles is a trivial feasible solution for both problems. Hereafter, we assume that $\mathcal{R} \subseteq \mathcal{C}$. Our next goal is to develop two efficient approaches for solving RRP (or RRP^U).

3. Approximation Algorithms for RRP

RRP is strongly NP-hard and, therefore, no polynomial-time exact algorithm can exist unless $\mathbf{P} = \mathbf{NP}$. In practice too, the size of the candidate set of roles $\mathcal{C}$ is typically in the thousands (or even more), and every subset of $\mathcal{C}$ could be a potential solution to RRP. Thus, even with 1,000 candidate roles, the total number of potential solutions exceeds 10^{301}, rendering complete enumeration an impractical approach to solve RRP. In this section, we propose two polynomial-time approximation algorithms. RRP is a generalization of the classical set-cover problem with the following differences: First, instead of a single set that needs to be covered in a set cover, the existing role system $\mathcal{R}$ contains multiple nondisjoint roles that each need to be covered in RRP. Second, an *exact cover* is emphasized, i.e., each role in $\mathcal{R}$ should be exactly the union of some sets in the new role system $\mathcal{S}$. Third, a role in $\mathcal{S}$ can be used to cover multiple roles in $\mathcal{R}$.

Motivated by the connection as well as the differences between RRP and the set cover problem, we now generalize two well-known polynomial-time approximation algorithms for the set-cover problem: the greedy algorithm (Chvatal 1979) and the randomized rounding algorithm (Raghavan and Thompson 1987). Compared to the set-cover problem, where only a single target set needs to be covered, RRP considers a more general and complicated many-to-many relationship between (target) multiple, nondisjoint roles (as shown in §4, the number of target roles in our real data is in the thousands, and these roles overlap significantly) and candidate roles. Naturally, this requires that we appropriately generalize both algorithms. Accordingly, the arguments for the validity of the theoretical worst-case bounds (in Theorems 1 and 2) as well as the time complexities of both algorithms reflect this generalization.

Although the practical performance of our algorithms is discussed in §4, the purpose of establishing their theoretical worst-case bounds is twofold: (1) to show that the solutions delivered by the algorithms cannot be arbitrarily bad (relative to the optimum) on *any* class of instances, and (2) to establish that the bounds are indeed the best that can be achieved by any efficient (i.e., polynomial time) algorithm for RRP. As is typically the case, the practical performance of our algorithms is much better than the provable, worst-case bounds: as we will see in §4, the solutions on instances obtained from real-world data are, on average, within 5.5% of their respective optima.

3.1. A Greedy Algorithm

Given an instance $\{\mathcal{R}, \mathcal{C}, c\}$ of RRP, we construct a weighted bipartite graph $B(\mathcal{R}, \mathcal{C}, \mathbf{E})$ as follows: One side of the partition consists of a node for each existing role $R_i \in \mathcal{R}$. The other side of the partition consists of a node for each candidate role $C_j \in \mathcal{C}$. The edge set $\mathbf{E}$ is constructed as follows: an edge $e(R_i, C_j) \in \mathbf{E}$ if and only if $C_j \subseteq R_i$. If $e(R_i, C_j)$ exists, then we refer to the sets R_i and C_j as *neighbors*. The weight of edge $e(R_i, C_j)$ is $w_{ij} = |R_i \cap C_j|$. The sum of the weights of all the edges incident on node C_j is denoted by $w_j = \sum_{e(R_i, C_j)\in E} w_{ij}$.

Consider a candidate role $C_j \in \mathcal{C}$. Clearly, C_j can only be (possibly) used to cover its direct neighbors (which are its supersets) in the graph B, otherwise the exact cover condition is violated. The edge weight w_{ij} indicates the number of permissions in R_i that can be covered by C_j. Thus, $w_j = \sum_{e(R_i, C_j)\in E} w_{ij}$ is a measure of the effectiveness of C_j in covering the existing roles in $\mathcal{R}$. Since the cost of C_j is $c(C_j)$, a measure of its cost per unit coverage is the ratio $c(C_j)/w_j$.

The greedy algorithm for RRP works as follows: iteratively pick a candidate with the lowest cost per unit coverage from $\mathcal{C}$, remove the permissions covered by this candidate from $\mathcal{R}$, and update the bipartite graph and cost per unit coverage values for the remaining sets in $\mathcal{C}$. We continue until all the permissions in $\mathcal{R}$ have been removed. Note that in the update after each iteration, those updated candidate roles $C_j \in \mathcal{C}$ with $w_j = 0$ (i.e., a candidate role consisting of permissions that have all been covered in earlier iterations, for each of its neighboring roles that requires some of these permissions) can be discarded from further consideration.

Algorithm 1 (Greedy).

```
1. $\mathcal{S} \leftarrow \varnothing$
2. **while** $\mathcal{R} \neq \{\varnothing\}$ **do**
     Find the candidate, say $C$, with the least cost per
         unit coverage in $\mathcal{C}$, resolving ties arbitrarily.
     Update $\mathcal{S}$: $\mathcal{S} \leftarrow \mathcal{S} \cup \{C\}$.
     Update $\mathcal{R}$: for each direct neighbor, say $R$, of $C$,
         we let $R \leftarrow R - C$.
     Update the edges and their weights in the
         bipartite graph $B$, based on the updated $\mathcal{R}$.
3. Output $\mathcal{S}$.
```

We now analyze the performance of Greedy. We start by establishing the following basic result. The proofs of all the technical results are provided in the online appendix (available as supplemental material at http://dx.doi.org/10.1287/ijoc.2014.0603).

Lemma 1. *If a feasible solution exists for RRP, then Algorithm* 1 *generates one such solution in a finite number of iterations.*

We now proceed with our analysis of the performance of Algorithm 1. For a given instance of RRP, we denote an optimal solution and its cost by OPT and OPT^c, respectively. Also, let the cost of the solution obtained by the algorithm be denoted by GRD^c. At the beginning of iteration t of the algorithm, let the cardinality (i.e., the number of permissions) of each (updated) role $R_i \in \mathcal{R}$ be m_i^t and let the total cardinality of (the updated system) $\mathcal{R}$ be $M(t) = \sum_{i=1}^{k} m_i^t$. Thus, at the beginning of iteration 1, $M = M(1) = \sum_{i=1}^{k} m_i^1$ is the sum of the cardinalities of all the roles in $\mathcal{R}$. Assume that candidate C_j is picked in iteration t by the algorithm. Then, we know that a total of w_j (the updated value after $t-1$ iterations, similar hereafter) permissions are covered (including duplicates across roles, if any) by C_j at a cost of $c(C_j)$. Thus, the cost per unit coverage $c(C_j)/w_j$ can be considered as the *price* for each of the w_j permissions C_j covers. Consequently, the cost of $c(C_j)$ is the sum of prices of the permissions it covers. Let us index the M permissions in the order in which they are covered by the iterations of the algorithm, resolving ties arbitrarily, to get a sequence of permissions $e_1, e_2, \ldots, e_M$, with respective prices $p_1, p_2, \ldots, p_M$. Then, $GRD^c = \sum_{i=1}^{M} p_i$. Since (1) the algorithm picks, in each iteration, the candidate with the lowest cost per unit coverage and (2) the cost per unit coverage of each candidate role is nondecreasing as the algorithm proceeds, we have that $p_i \le p_{i+1}$, $i = 1, 2, \ldots, M$. Consequently, $p_i \le (\sum_{j=i}^{M} p_j)/(M - i + 1)$. This implies the following result.

Theorem 1. *Algorithm 1 is $O(\ln M)$-factor, polynomial-time algorithm for RRP.*

The $O(\ln M)$-factor approximation guarantee offered for RRP by Algorithm 1 is the best possible for any polynomial-time algorithm, unless $\mathrm{P} = \mathrm{NP}$. As mentioned earlier, in the extreme case when the existing role system $\mathscr{R}$ consists of a single role, say $\mathscr{R} = \{R\}$, RRP degenerates into the classical set-cover problem. Also, in this special case, we have $M = |R|$. The following inapproximability result is well known for the set-cover problem: *there cannot exist a polynomial-time algorithm with an approximation guarantee better than $O(\ln M)$, unless* $\mathbf{P} = \mathbf{NP}$; see, e.g., Alon et al. (2006), Theorem 7. Also, again using the special case of the set-cover problem, it is easy to construct examples where Algorithm 1 achieves its worst-case bound of $O(\ln M)$. Thus, the bound in Theorem 1 is tight.

3.2. A Randomized Rounding Algorithm

This algorithm is based on rounding an optimal solution of the LP relaxation corresponding to a specific IP formulation for RRP. Accordingly, we will start by first describing the IP formulation.

As before, consider an instance $\{\mathscr{R}, \mathscr{C}, c\}$ of RRP. The decision variables are $z_j \in \{0, 1\}$, $j = 1, 2, \ldots, n$, defined as follows: $z_j = 1$ if candidate role $C_j \in \mathscr{C}$ is chosen in the solution $\mathscr{S}$; 0 otherwise. Recall from §3.1 that M is the sum of cardinalities (i.e., the number of permissions) of all the roles in $\mathscr{R}$. Let us index the M permissions arbitrarily as $e_1, e_2, \ldots, e_M$. Consider a permission e_ℓ, belonging, say, to role $R_i \in \mathscr{R}$. A candidate role $C \in \mathscr{C}$ can cover permission e_ℓ if and only if it meets the following two requirements simultaneously: (1) $e_\ell \in C$ and (2) $C \subseteq R_i$. Note that $\mathscr{C}$ may contain multiple candidate roles that can each cover permission e_ℓ. Let us denote the set of the indices of such candidates by $I(e_\ell)$. That is, $I(e_\ell) = \{j \mid e_\ell \in C_j,\ C_j \subseteq R_i,\ j \in \{1, 2, \ldots, n\}\}$. RRP requires that, for each permission in $\mathscr{R}$, at least one candidate role that can cover this permission is picked in the solution $\mathscr{S}$. We therefore have the following IP formulation for RRP:

$$\begin{aligned} &\text{Minimize} && \sum_{j=1}^{n} c(C_j) z_j \\ &\text{subject to:} && \sum_{j \in I(e_\ell)} z_j \ge 1, \quad \ell = 1, 2, \ldots, M \\ &&& z_j \in \{0, 1\}, \quad j = 1, 2, \ldots, n. \end{aligned} \qquad \text{(IP)}$$

Let OPT^c denote the optimal cost of this IP. The LP relaxation corresponding to the IP, which can be efficiently solved, is obtained using the constraints $0 \le z_j \le 1$ instead of $z_j \in \{0, 1\}$, for $j = 1, 2, \ldots, n$. Let us denote the optimal solution of the LP relaxation by $\bar{Z} = \{\bar{z}_1, \bar{z}_2, \ldots, \bar{z}_n\}$ and the corresponding objective function value by LP^c. Clearly, $LP^c \le OPT^c$.

The randomized rounding procedure now works as follows. For each candidate role $C_j \in \mathscr{C}$ independently, we include C_j with probability $\bar{z}_j$ in a potential solution $\bar{\mathscr{S}}$. Then, the expected cost of $\bar{\mathscr{S}}$ is

$$\begin{aligned} \mathbf{E}[\text{cost}(\bar{\mathscr{S}})] &= \sum_{j=1}^{n} c(C_j) \mathbf{Prob}[C_j \text{ is picked}] \\ &= \sum_{j=1}^{n} c(C_j) \bar{z}_j = LP^c \le OPT^c. \end{aligned}$$

Thus, the expected cost of the role system $\bar{\mathscr{S}}$ is at most the optimal cost OPT^c. Note, however, that $\bar{\mathscr{S}}$ may not be a feasible solution to (IP). We now obtain an upper bound on the probability that a permission e_ℓ is not covered by $\bar{\mathscr{S}}$. Let $\gamma = |I(e_\ell)|$. The permission e_ℓ is not covered by $\bar{\mathscr{S}}$ if and only if none of the candidate roles C_j, $j \in I(e_\ell)$, are included in $\bar{\mathscr{S}}$. The total number of these candidate roles is γ and their probabilities of being picked are determined by $\bar{Z}$. Without loss of generality, let $I(e_\ell) = \{1, 2, \ldots, \gamma\}$. Then,

$$\mathbf{Prob}[e_\ell \text{ is not covered by } \bar{\mathscr{S}}] = (1 - \bar{z}_1)(1 - \bar{z}_2) \ldots (1 - \bar{z}_\gamma).$$

The constraints of (IP) guarantee that $\bar{z}_1 + \bar{z}_2 + \cdots + \bar{z}_\gamma \ge 1$. Therefore, this probability is maximized when $\bar{z}_1 = \bar{z}_2 = \cdots = \bar{z}_\gamma = 1/\gamma$. Therefore,

$$\mathbf{Prob}[e_\ell \text{ is not covered by } \bar{\mathscr{S}}] \le \left(1 - \frac{1}{\gamma}\right)^{\gamma} \le \frac{1}{e}.$$

We independently generate $\lceil 2 \ln M \rceil$ such potential solutions and denote these by $\bar{\mathscr{S}}_\theta$, $\theta = 1, 2, \ldots, \lceil 2 \ln M \rceil$. Finally, we propose $\mathscr{S} = \bigcup_{\theta=1}^{\lceil 2 \ln M \rceil} \bar{\mathscr{S}}_\theta$ as a solution to RRP.

Algorithm 2 (Randomized Rounding).

```
1. $\mathscr{S} \leftarrow \varnothing$
2. Independently generate $\lceil 2 \ln M \rceil$ potential solutions
      to RRP: $\bar{\mathscr{S}}_1, \bar{\mathscr{S}}_2, \ldots, \bar{\mathscr{S}}_{\lceil 2 \ln M \rceil}$
3. $\mathscr{S} \leftarrow \bigcup_{\theta=1}^{\lceil 2 \ln M \rceil} \bar{\mathscr{S}}_\theta$
4. Output $\mathscr{S}$.
```

The following result establishes the performance guarantee of Algorithm 2.

Theorem 2. *Algorithm 2 is a polynomial-time algorithm for RRP that achieves the following two properties*: (i) *Its output $\mathscr{S} = \bigcup_{\theta=1}^{\lceil 2 \ln M \rceil} \bar{\mathscr{S}}_\theta$ is a feasible solution to RRP with high probability for sufficiently large M. In particular, the probability that $\mathscr{S}$ is feasible approaches 1 as $M \to \infty$.* (ii) *The expected cost of $\mathscr{S}$ is $O(\ln M)$ times the cost of the optimal solution.*

As we will see in §4, the value of M in the real data we use in our computations is more than 300,000. For each of the instances in our test bed, Algorithm 2 provided a feasible solution to RRP.

Remark on Derandomization. In theory, Algorithm 2 provides only an asymptotic guarantee of a feasible solution. It is possible to derandomize this algorithm in polynomial time, i.e., turn it into a deterministic algorithm so that it obtains a solution that is guaranteed to be feasible and has a cost that is guaranteed to be within an $O(\ln M)$ factor of the optimal cost. For brevity, we do not present this procedure here. For details on the various derandomization techniques, we refer the reader to Vazirani (2001).

4. Computational Experience

In this section, we test the algorithms developed in §3 for RRP and its variant RRP^U on instances based on a real data set obtained from a firm's ERP system. The main contributions of our computational work are threefold:

1. RRP is a practical problem arising from the need to effectively manage a firm's RBAC system. Using a role system that is currently being used in a firm's ERP implementation, we demonstrate the actual benefit of performing role refinement in a real-world context.
2. As established in §3, both Greedy and Randomized Rounding have the best possible theoretical performance guarantees and attractive time complexities. To supplement these results, we show that these two algorithms indeed provide good solutions on a real-world role refinement problem in acceptable time.
3. The two role refinement problems, RRP and RRP^U, although theoretically equivalent, have some differences in their practical usage (§2.2). We compare their relative cost reduction capabilities on the instances in our test bed.

Recall that both Greedy and Randomized Rounding offer the same (logarithmic factor) theoretical guarantee. In our experiments (reported later in this section), however, we have observed that Randomized Rounding consistently offers a better solution. It should be noted that this algorithm is more sophisticated, in the sense that it requires access to a good LP solver. Greedy, in contrast, is very simple to implement. Thus, if a good LP solver is available, then we recommend that Randomized Rounding be used. Otherwise, Greedy can be adopted.

4.1. Real Data

The data for generating our test bed were obtained from a firm's ERP system and include the entire role system—the list of roles and the permissions contained in each role—that is currently in use. A confidentiality agreement prevents us from disclosing the firm's identity.

The data were preprocessed via the following two steps:

- The data contained some duplicate roles that differed only in their names but contained an identical set of permissions. Upon inquiry, two reasons were provided for such occurrences: (1) a system administrator intended to create a new role by copying and revising an existing role, but no change was made and the new role simply continued to exist in the system, and (2) in some cases, a role is used by several types of users (e.g., users in different divisions), and an administrator may duplicate such a role with different, identifiable names for easier management. In either case, we removed such duplications.
- In the original role system, there may exist groups of permissions that always occur together. As a simple example, suppose permissions "a," "b," and "c" always occur together whenever they are included in a role. In this case, "abc" can be treated as a "composite" permission. A composite permission is *maximal* if we cannot add any other permission to it without losing the property of the individual permissions in the composite permission always occurring together. More formally, a set of permissions E is a maximal "composite" permission if it has the following properties: (1) $|E| \geq 2$; (2) for any two permissions $P_i, P_j \in E$, we have $P_i \in R \iff P_j \in R$, where R is any role in the original role system; and (3) if any other permission is added to E, then it no longer has property (2). As far as RRP is concerned, such a group of permissions is equivalent to a single "composite" permission and was therefore treated as such.

After preprocessing, the role system consists of 5,527 unique roles and 14,813 unique permissions, along with the corresponding permission-role assignment. Table 1 lists some basic statistics about the role system. As we can see, the sum of the cardinalities of the roles is 322,754 (this is the value of the constant M defined in §3.1), whereas the total number of unique permissions is only 14,813. This indicates a strong overlap between the permissions across the different roles.

4.2. Computational Experience with RRP

The following three characteristics of an arbitrary subset of roles from the real data set helped us create several smaller instances of RRP for testing purposes.

- *Size of the Role Subsystem*: The number of roles in the chosen subset of roles.
- *Ratio of Permissions to Roles*: Consider the set $\mathcal{P}$ of all the permissions in a firm's RBAC system. In theory,

Table 1 Basic Statistics for the Real-World Role System

Total number of roles	5,527
Sum of cardinalities (number of permissions) of roles	322,754
Total number of unique permissions	14,813
Average number of permissions in a role	58.40
Minimum number of permissions in a role	1
Maximum number of permissions in a role	1,675
Standard deviation of the number of permissions in a role	108.03

any subset of $\mathcal{P}$ could potentially be a role. In reality, however, a role is typically created to represent a combination of certain permissions that together constitute a business process (§2.1). On one hand, for a firm conducting relatively routine business processes and operating in a stable environment, a specific number of roles can fulfill its needs in most cases. On the other hand, a firm operating in a turbulent, highly competitive market typically needs to adopt a more agile style with frequent adjustments to business processes and, correspondingly, a frequent role-revision policy. Accordingly, the ratio of the total number of distinct permissions to the total number of roles in the role system is intended to capture a firm's operating style, with a relatively high value of this ratio suggesting more stable operations.

• *Standard Deviation of Permission Frequency*: The frequency of a permission is simply the number of roles this permission is part of. Clearly, permissions with higher frequency indicate that it is more commonly required by roles. A low variance in these frequencies suggests that the permissions are distributed evenly across roles. For a firm with a decentralized organizational structure, one would expect a relatively low variance reflecting a wider dispersal of decision-making and business functions.

4.2.1. Test Bed of RRP. As defined in §2.1, an instance of RRP for a firm is a triplet $\{\mathcal{R}, \mathcal{C}, c\}$, where $\mathcal{R}$ is the existing role system, $\mathcal{C}$ is the set of candidates for potential roles in the new role system, and c is the cost function. We now present our approach of generating instances of RRP.

Instances with Large Role Systems. We generated 32 role systems by randomly drawing 3,000 roles from the 5,527 roles in the data set. For a meaningful comparison across these instances, we retained only those random draws with similar values of the other two characteristics described above, namely Ratio of Permissions to Roles and Standard Deviation of Permission Frequency. The values of these two characteristics across the 32 role systems are quite stable, ranging from 4.2 to 4.4 for the former and from 32.0 to 35.9 for the latter. The sum of cardinalities (i.e., number of permissions) of the roles in each of the generated role systems is around 175,000.

The second component of a RRP instance, i.e., the candidate set $\mathcal{C}$ of new roles, would typically be determined by several requirements, including the firm's business needs, separation of duties, etc. Since such information was not available to us, we adopted a simple approach to generate a candidate set based on the reasonable assumption that a set of permissions is a "useful" candidate role only if it is a subset of at least one role in the existing role system. We generate the following five sets of candidate roles. The union of these five sets is our candidate set of roles.

1. $\mathcal{C}_1$: *"Composite" Permissions Identified by Preprocessing*. All maximal composite permissions identified in preprocessing are included in the candidate set of roles.

2. $\mathcal{C}_2$: *The Roles in the Original Roles*. Including all the original roles in the candidate set of roles guarantees two properties: (1) At least one feasible solution exists for the role refinement problem. (2) The refined role system will be at least as good as the original role system.

3. $\mathcal{C}_3$: *Pairwise Intersections of Original Roles*. The intersection $R_i \cap R_j$ is the subset of permissions common to the two original roles R_i and R_j and therefore has the potential to be used to cover both of them and possibly other roles. We generate all pairwise intersections in the original role system and include them as candidate roles.

4. $\mathcal{C}_4$: *Higher-Order Intersections of Original Roles*. Similar to the pairwise intersections, the intersections of three or more original roles could also be good potential candidate roles. However, it would not be feasible to include all of them in the candidate set. Consider, for example, an original role system with only 500 roles. Then the total number of higher order intersections is about $2^{500} - \binom{500}{2} - 500 - 1 = 3.3 \times 10^{150}$. Even checking this number of intersections would take an extraordinary amount of time. Furthermore, if we could somehow list all of them, the time and memory space required to solve the resulting LP relaxation in Algorithm 2 will be overwhelming. Therefore, to ensure tractability, we include a limited number of higher-order intersections in the candidate set. Specifically, for $n = 3, 4, \ldots, 10$, we generate n intersections by enumerating 40,000 randomly chosen trails. To clarify, consider $n = 3$ as an example. We randomly select three roles from the original set of roles and find their intersection. If this intersection is nonempty, then we include this intersection as a candidate role. This step is repeated 40,000 times. Then the entire process is repeated for $n = 4, 5, \ldots, 10$.

5. $\mathcal{C}_5$: *Candidate Roles from Maximal Bicliques*. Permissions and roles generate a natural bipartite graph: The nodes on one side of the bipartition correspond to permissions, and the nodes on the other side correspond to sets of roles. Each permission is connected to all the roles that contain this permission. Consider the bipartite graph $G(\mathcal{P}, \mathcal{R}, \mathbf{E})$, where $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ is the set of all permissions, $\mathcal{R} = \{R_1, R_2, \ldots, R_k\}$ is the set of all original roles, and edge $(P_i, R_j) \in \mathbf{E} \iff P_i \in R_j$. To illustrate, Figure 4 shows a bipartite graph consisting of five permissions and six roles. A *biclique* of G is a set of permissions $B_1 \subseteq \mathcal{R}$ and a set of roles $B_2 \subseteq \mathcal{P}$ such that each permission in B_1 is connected to all the roles in B_2. That is, edge $(P_i, R_j) \in \mathbf{E}$, $\forall\, P_i \in B_1$, and $R_j \in B_2$. A biclique $\{B_1, B_2\}$ is *maximal* if there exists no biclique $\{B_1^*, B_2^*\}$ such that $\{B_1 \cup B_2\} \subset \{B_1^* \cup B_2^*\}$. We find

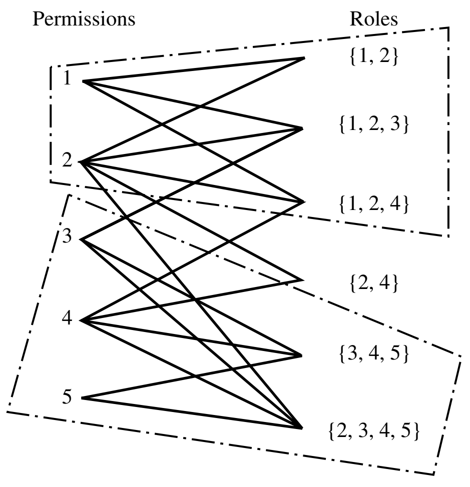


Figure 4 **Role Refinement: Illustrating Maximal Bicliques in the Permission-Role Bipartite Graph**

all maximal bicliques and include the sets of permissions in them as candidate roles. Figure 4 illustrates two maximal bicliques by dotted lines. The set of permissions in one of these two bilcliques is $\{1, 2\}$; it is $\{3, 4, 5\}$ in the other.

The candidate set of roles is $\mathscr{C} = \bigcup_{i=1}^{5} \mathscr{C}_i$. For each of the 32 role systems, the cardinality of the corresponding candidate sets was typically around 75,000.

We assume that the cost of a candidate role includes a fixed part and a variable part; i.e., $c(C_i) = c^{\text{fix}} + c^{\text{var}}(C_i)$. The fixed cost is the same for each role and reflects the common system resource expenses and administrative cost to manage a role, whereas the variable cost captures the differences in these costs caused by the specific sets of permissions different roles contain. Since we could not differentiate the underlying operational complexities of different permissions due to the lack of such information, we make the following reasonable assumption: The marginal cost of adding a new permission to a role increases with the size (i.e., number of permissions) of the role, since one has to account for the possible interactions of the new permission with the existing permissions in the role. This immediately implies a cost function that is convex in the number of permissions in a role. For simplicity, we used a quadratic form:

$$c(C_i) = c^{\text{fix}} + k_1|C_i| + k_2|C_i|^2,$$

where $|C_i|$ denotes the cardinality of C_i. The second term is the variable cost associated with managing permissions individually in the role, which is proportional to the total number of permissions. The third term represents the joint variable cost of collectively managing all permissions in the role. In our experiments, we adopt four sets of values for the cost parameters $(c^{\text{fix}}, k_1, k_2)$: $(1, 0, 0)$, $(1, 0.5 \times 10^{-2}, 0.5 \times 10^{-5})$, $(1, 10^{-2}, 10^{-5})$, and $(1, 2 \times 10^{-2}, 2 \times 10^{-5})$. The first set of parameter values corresponds to the situation where we are only interested in minimizing the number of roles in the refined system. The other three sets of parameters impose a penalty on the size of the roles: higher values of k_1 and k_2 imply a harsher penalty on the size of a role.

The values of the parameters k_1 and k_2 in our cost function determine the relative importance between the cardinality of the role system and the sizes of its roles in the optimization problem. Note, however, that the input characteristics of a specific instance, e.g., the total number of users and the total number of permissions, naturally influence (either directly or indirectly) both the number of roles and their sizes. Thus, a more sophisticated approach would be to endogenize the values of k_1 and k_2, i.e., decide these values automatically for an instance, based on the input characteristics of the instance. We suggest this approach as a potential direction for future work.

Instances with Small Role Systems. To demonstrate the robustness of the role refinement scheme, we also generate role systems containing n roles, where $n \in \{600, 700, 800, \ldots, 1{,}800\}$, in addition to the relatively large role systems with $n = 3{,}000$ roles. We generate 32 role systems for each value of n. To ensure that our computations are grounded in real RBAC practice, the instances with role systems containing n roles are generated by randomly drawing n roles from the real-world data. We use the same approach as previously described for generating the candidate sets for these smaller role systems, and use the same quadratic form of the cost function and the four sets of values for the parameters $(c^{\text{fix}}, k_1, k_2)$.

Together, the combination of the different values of n and $(c^{\text{fix}}, k_1, k_2)$ provides a total of $14 \times 32 \times 4 = 1{,}792$ instances of RRP for the test bed. We summarize the results of applying role refinement to these instances in §4.2.2.

4.2.2. Computational Results of RRP.

All computations were executed on an Intel Core computer with 3.40 GHz CPU and 8 GB RAM. The LP relaxation required for Algorithm 2 (see §3.2) was solved using ILOG CPLEX (version 11.1).

Tables 2 and 3 present the results for the instances of RRP with 3,000 roles. Table 2 presents the results for the 32 instances with cost parameters $(c^{\text{fix}}, k_1, k_2) = (1, 10^{-2}, 10^{-5})$. Given that role refinement is not a daily activity, we note that the CPU time required by the algorithm is reasonable for each of the 32 instances. The refined role systems offer significant cost reductions ranging between 6.8% and 12.0%, with an average reduction of 8.8%. Table 3 summarizes the results for the other cost parameters.

A natural question arises: how good are these reductions relative to the best possible? The optimal cost of the LP relaxation corresponding to the IP formulation

Table 2 Computational Results for the 32 Instances of RRP with 3,000 Roles and Cost Parameters $(c^{\text{fix}}, k_1, k_2) = (1, 10^{-2}, 10^{-5})$

Instance	Cost of existing role system	Cost of new role system	Cost reduction (%)	Gap$^{\text{AL}}$ (%)	New cardinality	Original granularity	New granularity	CPU time (sec.)
1	5,180.0	4,647.2	10.3	2.6	3,005	58.3	44.8	518
2	5,114.1	4,656.0	9.0	1.6	2,986	56.8	45.5	363
3	5,030.8	4,646.7	7.6	2.2	3,016	54.3	43.8	361
4	5,079.0	4,572.9	10.0	1.4	2,991	55.9	43.5	374
5	5,262.3	4,805.1	8.7	0.9	2,996	59.3	47.9	449
6	5,218.0	4,792.0	8.2	1.8	3,006	58.3	47.1	457
7	5,257.4	4,767.7	9.3	0.7	3,009	59.4	46.9	445
8	5,210.2	4,840.9	7.1	1.8	3,019	58.6	48.2	453
9	5,168.3	4,803.4	7.1	2.7	3,029	56.4	45.8	371
10	5,033.6	4,618.3	8.2	0.6	2,975	54.6	44.6	385
11	5,199.7	4,774.3	8.2	2.0	3,013	57.6	46.3	377
12	5,042.3	4,697.5	6.8	1.0	3,001	54.4	45.1	373
13	5,245.4	4,722.3	10.0	2.6	3,018	59.7	45.8	455
14	5,384.6	4,823.4	10.4	1.6	3,008	62.4	48.1	522
15	5,421.0	4,772.2	12.0	0.6	2,981	62.9	47.6	466
16	5,309.5	4,795.5	9.7	2.2	3,006	61.2	48.2	474
17	5,013.2	4,640.4	7.4	2.4	3,022	54.3	43.6	360
18	5,063.9	4,675.8	7.7	2.3	3,008	55.5	45.1	407
19	5,067.5	4,684.9	7.6	2.6	3,014	55.6	45.0	363
20	5,063.7	4,689.5	7.4	2.6	2,994	55.5	45.9	366
21	5,385.4	4,884.9	9.3	2.2	3,020	62.1	49.1	536
22	5,319.6	4,862.8	8.6	2.0	3,026	60.5	47.8	474
23	5,306.6	4,821.3	9.1	1.7	2,986	60.2	48.4	426
24	5,460.1	4,851.9	11.1	2.0	3,015	62.7	47.3	482
25	5,265.8	4,710.5	10.5	1.5	2,995	59.6	45.9	465
26	5,139.1	4,616.9	10.2	0.6	3,003	57.4	44.1	394
27	5,248.2	4,672.3	11.0	1.1	2,977	59.4	45.7	477
28	5,247.6	4,758.6	9.3	1.2	3,006	59.1	46.6	405
29	5,241.2	4,866.9	7.1	2.4	3,038	58.3	47.0	425
30	5,176.9	4,744.0	8.4	1.9	3,021	57.5	45.7	407
31	5,157.3	4,776.1	7.4	0.8	2,997	57.0	47.0	384
32	5,231.9	4,827.8	7.7	1.7	3,006	58.2	47.5	405
Average	5,204.5	4,744.4	8.8	1.7	3,005.8	58.2	46.3	425.6

of an instance of RRP (see (IP), §3.2) is a lower bound on the optimal cost of that instance. Thus, the quantity

$$\text{Gap}^{\text{AL}} = \frac{\text{Cost of the new role system}}{\text{Optimal cost of the LP relaxation}} - 1$$

is a measure of the near optimality of the solutions obtained by Algorithm 2. Over the 32 instances of RRP in Table 2, the average value of Gap$^{\text{AL}}$ is 1.73%, establishing that Randomized Rounding indeed offers high-quality solutions.

The overall objective of RRP is to minimize the total cost of the refined role system. However, we are also interested in examining the impact on two characteristics of the refined role system: *cardinality* and *granularity*. The cardinality of a role system is the number of roles it contains, and its granularity is the average size of a role (i.e., the average number of permissions in a role) in the role system. The values of these two characteristics are also included in the results. Over the 32 instances of RRP in Table 2, the average granularity of the original systems is 58.2 and this number is 46.3 for the refined role systems. Thus, the shortened role is a significant source of cost reduction.

In RRP, there is a natural tradeoff between the granularity and the cardinality of the refined role system: As the average size of a role increases, the number of roles reduces, and vice versa. This tradeoff is influenced by

Table 3 Computational Results for Instances of RRP with 3,000 Roles Summarized by Cost Parameters

Cost parameters	Cost of existing role system	Cost of new role system	Cost reduction (%)	Gap$^{\text{AL}}$ (%)	New cardinality	Original granularity	New granularity	CPU time (sec.)
$(1, 0, 0)$	3,000.0	2,928.2	2.4	0.6	2,928.2	58.2	54.0	283.6
$(1, 0.5\times10^{-2}, 0.5\times10^{-5})$	4,103.2	3,870.7	5.7	1.7	2,977.7	58.2	47.7	334.4
$(1, 10^{-2}, 10^{-5})$	5,204.5	4,744.4	8.8	1.7	3,005.8	58.2	46.3	425.6
$(1, 2\times10^{-2}, 2\times10^{-5})$	7,406.1	6,510.2	12.1	2.5	3,075.3	58.2	44.7	379.8

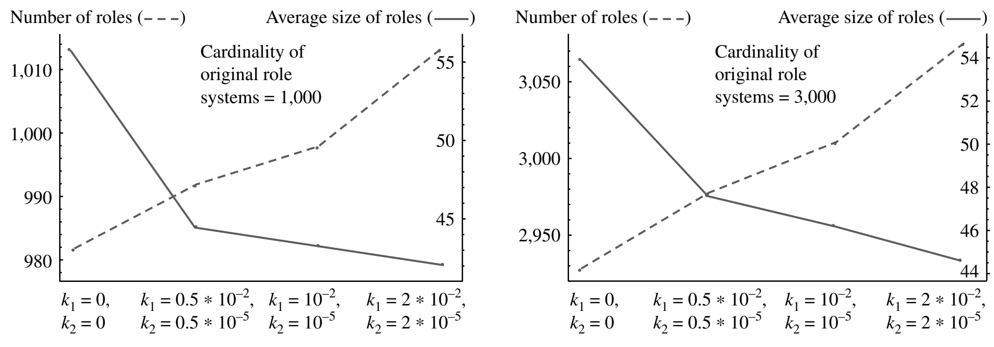


Figure 5 Role Refinement: Illustrating the Tradeoff Between Average Size of a Role (i.e., Granularity) and the Number of Roles

the cost function. Our role refinement scheme allows role managers to assign an arbitrary nonnegative cost to each candidate role, based on preference and the organization's operational environment. Generally speaking, if the cost associated with managing a bigger role is significantly higher than that with a smaller role, then role refinement tends to produce a new role system with a relatively high number of roles but smaller average size (i.e., higher granularity). In contrast, if the cost of managing a role does not increase much with its size, then the refined role system will have fewer roles and higher average size (i.e., lower granularity). To demonstrate this tradeoff, Table 3 summarizes the results of the instances of RRP with 3,000 roles for the four sets of cost parameters. Each row in this table shows the average of the values for the corresponding 32 instances. Figure 5 plots the average value of the number of roles and the average size of roles in the refined role system. We also plot these values for the instances of RRP with 1,000 roles. The tradeoff is clearly seen in these figures.

The results of the smaller instances in our test bed are presented in Tables 4–7, each of which corresponds to one of the four sets of values of the cost parameters $(c^{\text{fix}}, k_1, k_2)$. Each row in these tables shows the average values over the 32 instances with n roles, where $n \in \{600, 700, 800, \ldots, 1{,}800\}$. The results from these smaller instances of RRP demonstrate the robustness, effectiveness, and efficiency of the role refinement scheme and a similar tradeoff between cardinality and granularity in the refined role systems.

Remark on Repeated Application of the Algorithms. Since Randomized Rounding is a heuristic, applying the algorithm again on the refined role system has the potential to further improve the result. Our computations show that repeated application does deliver an additional cost reduction for some of the instances of RRP. However, the additional improvements are marginal: less than 1%. For the 32 instances in Table 2, the average additional percentage improvement (i.e., additional cost reduction) from applying this algorithm again on the refined role system is 0.42%. On five of 32 instances, the percentage improvement is 0. The highest additional percentage improvement is 1.17%. Further applications show no significant marginal improvements. The results are similar for Algorithm 1.

4.3. Computational Experience with RRP$^{\text{U}}$

Recall from §2.2 that the variant RRP$^{\text{U}}$ requires as input the set of permissions for each user. Since this information was deemed sensitive from a security viewpoint,

Table 4 Computational Results for Instances of RRP with n Roles, $n \in \{600, 700, 800, \ldots, 1{,}800\}$, and Cost Parameters $(c^{\text{fix}}, k_1, k_2) = (1, 0, 0)$

Original cardinality	Cost of existing role system	Cost of new role system	Cost reduction (%)	Gap$^{\text{AL}}$ (%)	New cardinality	Original granularity	New granularity	CPU time (sec.)
600	600.0	588.9	1.8	0.8	588.9	81.1	70.9	8.6
700	700.0	683.5	2.4	1.1	683.5	82.8	70.6	7.4
800	800.0	770.0	3.8	0.9	770.0	82.9	68.1	8.9
900	900.0	885.8	1.6	0.6	885.8	60.7	55.8	10.0
1,000	1,000.0	981.8	1.8	0.7	981.8	61.4	55.9	16.3
1,100	1,100.0	1,077.8	2.0	0.9	1,077.8	61.2	55.3	20.7
1,200	1,200.0	1,166.3	2.8	0.7	1,166.3	61.3	53.7	24.0
1,300	1,300.0	1,258.5	3.2	0.6	1,258.5	58.9	55.4	20.0
1,400	1,400.0	1,350.5	3.5	0.6	1,350.5	59.9	55.6	24.7
1,500	1,500.0	1,440.6	4.0	0.7	1,440.6	59.4	55.0	29.5
1,600	1,600.0	1,524.0	4.7	0.5	1,524.0	59.6	54.1	43.2
1,700	1,700.0	1,615.3	5.0	0.6	1,615.3	59.0	53.4	56.0
1,800	1,800.0	1,703.4	5.4	0.7	1,703.4	59.5	53.2	62.9

Table 5 **Computational Results for Instances of RRP with n Roles, $n \in \{600, 700, 800, \ldots, 1{,}800\}$, and Cost Parameters** $(c^{\text{fix}}, k_1, k_2) = (1, 0.5 \times 10^{-2}, 0.5 \times 10^{-5})$

Original cardinality	Cost of existing role system	Cost of new role system	Cost reduction (%)	Gap$^{\text{AL}}$ (%)	New cardinality	Original granularity	New granularity	CPU time (sec.)
600	913.8	820.2	10.2	1.2	597.4	81.1	59.3	9.4
700	1,073.9	947.2	11.8	1.5	693.1	82.8	58.4	8.9
800	1,228.0	1,061.5	13.5	1.6	781.4	82.9	57.2	9.2
900	1,239.7	1,141.0	8.0	0.9	893.9	60.7	44.9	12.2
1,000	1,382.2	1,263.7	8.6	1.3	991.9	61.4	44.6	18.4
1,100	1,518.0	1,382.0	9.0	1.5	1,089.4	61.2	43.8	23.6
1,200	1,657.5	1,488.8	10.2	1.3	1,178.4	61.3	43.0	29.3
1,300	1,766.3	1,651.3	6.5	1.3	1,270.7	58.9	50.5	27.5
1,400	1,911.2	1,780.1	6.9	1.3	1,363.8	59.9	51.3	36.5
1,500	2,043.3	1,887.1	7.6	1.4	1,455.1	59.4	50.0	42.3
1,600	2,183.4	1,989.5	8.9	1.1	1,540.3	59.6	49.2	71.2
1,700	2,311.0	2,105.5	8.9	1.5	1,633.2	59.0	48.9	84.5
1,800	2,453.6	2,221.8	9.4	1.6	1,725.6	59.5	48.7	100.0

Table 6 **Computational Results for Instances of RRP with n Roles, $n \in \{600, 700, 800, \ldots, 1{,}800\}$, and Cost Parameters** $(c^{\text{fix}}, k_1, k_2) = (1, 10^{-2}, 10^{-5})$

Original cardinality	Cost of existing role system	Cost of new role system	Cost reduction (%)	Gap$^{\text{AL}}$ (%)	New cardinality	Original granularity	New granularity	CPU time (sec.)
600	1,226.8	1,042.0	15.0	1.6	603.6	81.1	58.0	9.5
700	1,447.0	1,195.0	17.4	1.6	699.5	82.8	56.7	7.2
800	1,655.0	1,335.7	19.3	1.9	792.0	82.9	55.1	9.1
900	1,579.9	1,381.6	12.5	0.9	901.1	60.7	43.6	10.4
1,000	1,765.1	1,527.7	13.4	1.2	997.9	61.4	43.4	16.4
1,100	1,936.8	1,661.7	14.2	1.3	1,095.8	61.2	42.4	20.8
1,200	2,115.8	1,796.4	15.1	1.7	1,190.9	61.3	41.8	27.7
1,300	2,234.8	1,983.7	11.2	1.4	1,286.0	58.9	46.4	27.7
1,400	2,424.8	2,146.7	11.5	1.7	1,382.7	59.9	47.1	37.7
1,500	2,589.1	2,263.6	12.6	1.6	1,474.2	59.4	45.8	46.7
1,600	2,769.4	2,441.2	11.8	1.4	1,553.8	59.6	48.2	66.0
1,700	2,924.9	2,567.6	12.2	1.3	1,642.7	59.0	47.7	74.8
1,800	3,110.3	2,710.3	12.9	1.6	1,737.4	59.5	47.4	102.3

the firm could not share it with us. Nevertheless, using the fact that the set of permissions held by a user is the union of some subset of roles, we simulated a process for users to acquire permissions to generate a test bed of RRP$^{\text{U}}$.

4.3.1. Test Bed of RRP$^{\text{U}}$. As defined in §2.2, an instance of RRP$^{\text{U}}$ is a triplet $\{\mathcal{U}, \mathcal{C}, c\}$, where $\mathcal{U}$ is the set system of permissions of active users, $\mathcal{C}$ is the set of candidates for potential roles in the new role system, and c is the cost function.

Table 7 **Computational Results for Instances of RRP with n Roles, $n \in \{600, 700, 800, \ldots, 1{,}800\}$, and Cost Parameters** $(c^{\text{fix}}, k_1, k_2) = (1, 2 \times 10^{-2}, 2 \times 10^{-5})$

Original cardinality	Cost of existing role system	Cost of new role system	Cost reduction (%)	Gap$^{\text{AL}}$ (%)	New cardinality	Original granularity	New granularity	CPU time (sec.)
600	1,853.7	1,466.6	20.8	1.2	613.8	81.1	55.6	9.4
700	2,194.0	1,678.5	23.5	1.4	711.0	82.8	54.6	8.2
800	2,510.0	1,856.1	26.0	1.3	810.1	82.9	52.2	8.9
900	2,256.0	1,855.5	17.7	1.0	911.2	60.7	42.4	9.9
1,000	2,525.8	2,054.7	18.6	1.6	1,013.2	61.4	42.2	14.9
1,100	2,768.8	2,213.5	20.0	1.2	1,110.1	61.2	41.0	20.3
1,200	3,026.3	2,387.3	21.1	1.7	1,209.9	61.3	40.2	28.0
1,300	3,165.6	2,752.6	13.0	0.8	1,285.5	58.9	48.3	21.1
1,400	3,445.2	2,985.1	13.3	1.1	1,381.8	59.9	49.0	28.7
1,500	3,673.5	3,144.1	14.4	1.3	1,474.0	59.4	47.9	36.1
1,600	3,933.8	3,309.0	15.9	1.2	1,565.4	59.6	47.2	64.7
1,700	4,144.5	3,475.5	16.1	1.2	1,655.0	59.0	46.7	80.2
1,800	4,414.9	3,656.4	17.2	1.3	1,745.7	59.5	46.5	90.3

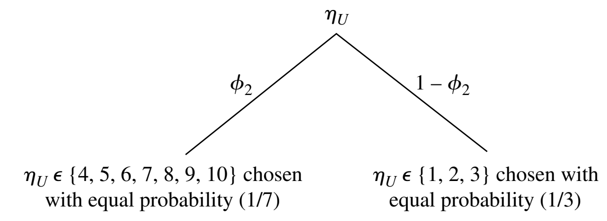

η_U
ϕ_2
$1-\phi_2$
$\eta_U \in \{4, 5, 6, 7, 8, 9, 10\}$ chosen with equal probability (1/7)
$\eta_U \in \{1, 2, 3\}$ chosen with equal probability (1/3)

Figure 6 Generating the Number of Roles a User Holds

Our communication with practitioners provided us with the following useful clues about user-permission assignments: (1) most users are assigned to a small number (typically 1, 2, or 3) of roles, and (2) roles normally represent specific business-related tasks; hence they are naturally partitioned into clusters that reflect a firm's organizational structure. The following scheme for generating user permissions incorporates the aforementioned guidance and uses three parameters ϕ_i, $i = 1, 2, 3$:

1. The number of users in $\mathcal{U}$ is $\phi_1 \in \{4{,}000, 5{,}000, 6{,}000\}$.
2. Denote the number of roles a user, say U, holds by η_U. To choose a value for η_U, we consider two cases: in case 1, with probability $\phi_2 \in \{0.1, 0.2\}$, $\eta_U = 4, 5, 6, 7, 8, 9$ or 10, with equal probability ($\frac{1}{7}$). In case 2, with probability $1-\phi_2$, $\eta_U = 1, 2$ or 3, with equal probability ($\frac{1}{3}$). Figure 6 depicts this process.
3. To generate user-permission assignments, we first partition the set of roles $\mathcal{R}$ evenly into five disjoint subsets: $\mathcal{R}_1$ through $\mathcal{R}_5$. Then, for user U, consider two cases: in case 1, with probability $\phi_3 \in \{0.1, 0.5, 0.9\}$, she is assigned to η_U roles randomly selected from $\mathcal{R}$. In case 2, with probability $1-\phi_3$, she is assigned to η_U roles randomly selected from a single subset R_i, chosen with equal probability from among the five possibilities. The set of permissions for U is then the union of the permissions in the roles she has been assigned to. Finally, the sets of permissions of all users constitutes the required set system $\mathcal{U}$.

For each of the 32 instances of RRP in Table 2 (§4.2.2), we generate 18 instances of RRP^U: one each for the 18 combinations of the three parameters ϕ_i, $i = 1, 2, 3$. Thus, the total number of instances of RRP^U in our test bed is $18 \times 32 = 576$. For an instance of RRP^U, we use the existing and refined roles (with the latter as obtained by Algorithm 2) in the corresponding instance of RRP as the set of candidate role. The cost function is as described in §4.2.1 with cost parameters $(c^{\text{fix}}, k_1, k_2) = (1, 10^{-2}, 10^{-5})$.

4.3.2. Computational Results of RRP^U. The instances of RRP^U were solved on the same platform as that for the RRP instances. The average CPU time required by Randomized Rounding to obtain the solution for an instance is 30.7 seconds. Across the 576 instances, the percentage of cost reduction ranges from 7.9% to 16.0%, with an average reduction of 11.4%. The average percentage of cost reduction corresponding to the three parameters ϕ_i, $i = 1, 2, 3$, is summarized in Table 8. Each cell in this table represents the average cost reduction over the corresponding 32 instances.

Table 8 Average Percentage Cost Reduction for Instances of RRP^U

	$\phi_2 = 0.1$ (%)			$\phi_2 = 0.2$ (%)		
	$\phi_3 = 0.1$	$\phi_3 = 0.5$	$\phi_3 = 0.9$	$\phi_3 = 0.1$	$\phi_3 = 0.5$	$\phi_3 = 0.9$
$\phi_1 = 4{,}000$	13.2	13.2	13.2	12.0	11.9	11.9
$\phi_1 = 5{,}000$	11.6	11.6	11.5	10.8	10.8	10.8
$\phi_1 = 6{,}000$	10.7	10.7	10.8	10.2	10.2	10.2

5. Summary and Discussion

Role-based access control is the access control approach used by most firms. The focus of this work is to reduce the administrative costs of a RBAC implementation. To this end, we develop two algorithms that are proved to provide good performance guarantees in polynomial time. Next, using a real data set collected from a firm's ERP implementation we demonstrate the viability of our solution approach. The approach solves real instances of the problem in a reasonable amount of time (less than 10 minutes) and yields an administrative cost reduction of about 10%. Finally, the solutions provided by our approach are of good quality, judging by the fact that these solutions are within about 3% of the lower bound on the optimal cost. We next discuss two issues that are of practical interest and could serve as the basis for future research on the role refinement problem.

5.1. Expansibility

Expansibility is a property that allows the current role system to accommodate future needs. These needs could arise because existing users require modifications in their respective sets of permissions or because new users join the firm with different job descriptions and hence require new sets of permissions. If the existing role system can accommodate future needs, then its life can be extended, at least for the purposes of operational feasibility. The two approaches (RRP and RRP^U) can be examined from the lens of expansibility. Because a solution to RRP is guaranteed to be able to express every role in the current role system, it follows that a RRP solution must be at least as expansible as the current role system. But such a claim cannot be made about a RRP^U solution: although such a solution will guarantee that the permissions of every current user are expressible, there is no ordinal statement that can be made comparing the expansibility of the current role system and that of a RRP^U solution. Thus, it is possible that a RRP^U solution is less expansible than the current role system. The RRP^U algorithm is less desirable on the dimension of expansibility, but its cost reduction (vis-a-vis the current role system) is typically greater than what is offered by the RRP algorithm. Thus, between RRP and RRP^U we have an expansibility-cost tradeoff. This tradeoff can be better understood with an extreme example. Consider a role system where every role consists of a single permission. Clearly, such a role system has the maximum possible expansibility; however, the cost of managing such a (single-permission) role system could be exorbitant. RRP and RRP^U represent two intermediate approaches on the expansibility-cost continuum; RRP favors expansibility at the expense of administrative cost, whereas RRP^U favors administrative cost at the expense of expansibility.

5.2. Dynamic Role Refinement

In this paper, we have addressed a static role refinement problem. However, role refinement is typically not a terminal solution, in the sense that the need for defining new roles arises continuously over time. Consequently, role refinement is also a recurring need. Although a role refinement typically reduces maintenance cost, it also incurs a fixed cost that includes the time and effort to update role definitions, user-role mappings, etc. Thus, to minimize cost over the lifetime of a RBAC system, the timings of the multiple role refinements need to be chosen carefully.

To this end, it may be convenient to consider a timeline $[0, T]$ of the RBAC system and view both the universe of users (i.e., their set system of permissions) $\mathcal{U}(t)$ and the role definitions $\mathcal{R}(t)$ as functions of time t, $t \in [0, T]$. The administrative cost $\Pi(t)$ at time t is a function of $\mathcal{R}(t)$ (i.e., $\Pi(t) = \Pi(\mathcal{R}(t))$). A role refinement at any time incurs a fixed cost ω. The goal would be to minimize the total cost, which is the sum of the administrative and fixed costs, over the useful system life T of the system.

Supplemental Material

Supplemental material to this paper is available at http://dx.doi.org/10.1287/ijoc.2014.0603.

References

Alon N, Moshkovitz D, Safra S (2006) Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms* 2(2):153–177.

Bai X, Nunez M, Kalagnanam JR (2011) Managing data quality risk in accounting information systems. *Inform. Systems Res.* 23(2):453–473.

Bai X, Gopal R, Nunez M, Zhdanov D (2012) On the prevention of fraud and privacy exposure in process information flow. *INFORMS J. Comput.* 24(3):416–432.

Basu A, Kumar A (2002) Research commentary: Workflow management issues in e-business. *Inform. Systems Res.* 13(1):1–14.

Botha RA, Eloff JHP (2001) Separation of duties for access control enforcement in workflow environments. *IBM Systems J.* 40(3): 666–682.

Chekuri C, Clarkson KL, Har-Peled S (2009) On the set multicover problem in geometric settings. *Proc. 25th Annual Sympos. Computational Geometry* (ACM, New York), 341–350.

Chvatal V (1979) A greedy heuristic for the set-covering problem. *Math. Oper. Res.* 4(3):233–235.

Colantonio A, Di Pietro R, Ocello A (2008) A cost-driven approach to role engineering. *Proc. 2008 ACM Sympos. Appl. Comput.* (ACM, New York), 2129–2136.

Colantonio A, Di Pietro R, Verde NV (2012) A business-driven decomposition methodology for role mining. *Comput. Security* 31(7): 844–855.

Colantonio A, Di Pietro R, Ocello A, Verde NV (2009a) A probabilistic bound on the basic role mining problem and its applications. *Proc. 24th IFIP TC 11 Internat. Inform. Security Conf., Cyprus Greece*.

Colantonio A, Di Pietro R, Ocello A, Verde NV (2009b) A formal framework to elicit roles with business meaning in RBAC systems. *Proc. 14th ACM Sympos. Access Control Models Tech.* (ACM, New York), 85–94.

Colantonio A, Di Pietro R, Ocello A, Verde NV (2010) Taming role mining complexity in RBAC. *Comput. Security* 29(5):548–564.

Colantonio A, Di Pietro R, Ocello A, Verde NV (2011) A new role mining framework to elicit business roles and to mitigate enterprise risk. *Decision Support Systems* 50(4):715–731.

Coleman K (2008) Separation of duties and IT security. Accessed July 1, 2014, http://www.csoonline.com/article/446017/separation-of-duties-and-it-security.

Coyne EJ (1995) Role-engineering. *1st ACM Workshop on Role-Based Access Control* (ACM Press, New York), 4.

Elliott A, Knight S (2010) Role explosion: Acknowledging the problem. *Proc. 2010 Internat. Conf. Software Engrg. Res. Practice, Las Vegas, NV*, 349–355.

Ene A, Horne W, Milosavljevic N, Rao P, Schreiber R, Tarjan RE (2008) Fast exact and heuristic methods for role minimization problems. *Proc. 13th ACM Sympos. Access Control Models Tech.* (ACM, New York), 1–10.

Ferraiolo DF, Kuhn DR (1992) Role-based access control. *15th National Comput. Security Conf., Baltimore*.

Ferraiolo DF, Barkley JF, Kuhn DR (1999) A role-based access control model and reference implementation within a corporate intranet. *ACM Trans. Inform. System Security* 2(1):34–64.

Ferraiolo DF, Cugini JA, Kuhn DR (1995) Role-based access control (RBAC): Features and motivations. *Proc. 11th Annual Comput. Security Appl. Conf., New Orleans*, 241–248.

Ferraiolo DF, Gilbert DM, Lynch N (1993) An examination of federal and commercial access control policy needs. *Proc. 16th NIST-NSA National Comput. Security Conf., Baltimore*, 107–116.

Ferraiolo DF, Kuhn DR, Chandramouli R (2007) *Role-Based Access Control*, 2nd ed. (Artech House, Norwood, MA).

Ferraiolo DF, Sandhu RS, Gavrila S, Kuhn DR, Chandramouli R (2001) Proposed NIST standard for role-based access control. *ACM Trans. Inform. System Security* 4(3):224–274.

Frank M, Basin D, Buhmann JM (2008) A class of probabilistic models for role engineering. *Proc. 15th ACM Conf. Comput. Comm. Security* (ACM, New York), 299–310.

Frank M, Buhmann JM, Basin D (2010) On the definition of role mining. *Proc. 15th ACM Sympos. Access Control Models Tech.* (ACM, New York), 35–44.

Frank M, Buhman J, Basin D (2013) Role mining with probabilistic models. *ACM Trans. Inform. System Security* 15(4):Article 15.

Guo Q, Vaidya J, Atluri V (2008) The role hierarchy mining problem: Discovery of optimal role hierarchies. *Comput. Security Appl. Conf.*, 237–246.

Hall NG, Hochbaum DS (1986) A fast approximation algorithm for the multicovering problem. *Discrete Appl. Math.* 15(1):35–40.

Hall NG, Hochbaum DS (1992) The multicovering problem. *Eur. J. Oper. Res.* 62(3):323–339.

Har-Peled S, Lee M (2012) Weighted geometric set cover problems revisited. *J. Computational Geometry* 3(1):65–85.

Karp RM (1972) Reducibility among combinatorial problems. Miller RE, Thatcher JW, eds. *Complexity Comput. Comput.* (Plenum Press, New York), 85–103.

Kuhlmann M, Shohat D, Schimpf G (2003) Role mining–revealing business roles for security administration using data mining technology. *Proc. 8th ACM Sympos. Access Control Models Tech.* (ACM, New York), 179–186.

Li N, Wang Q (2008) Beyond separation of duty: An algebra for specifying high-level security policies. *J. ACM* 55(3):Article 12.

Li N, Tripunitara MV, Bizri Z (2007) On mutually exclusive roles and separation-of-duty. *ACM Trans. Inform. System Security* 10(2):Article 5.

Lu H, Vaidya J, Atluri V (2008) Optimal Boolean matrix decomposition: Application to role engineering. *Proc. 2008 IEEE 24th Internat. Conf. Data Engrg.* (IEEE Computer Society, Washington, DC), 297–306.

Molloy I, Chen H, Li T, Wang Q, Li N, Bertino E, Calo S, Lobo J (2008) Mining roles with semantic meanings. *Proc. 13th ACM Sympos. Access Control Models Tech.* (ACM, New York), 21–30.

O'Connor AC, Loomis RJ (2010) Economic Analysis of role-based access control. Accessed July 1, 2014, http://csrc.nist.gov/groups/SNS/rbac/documents/20101219_RBAC2_Final_Report.pdf.

Osborn S, Sandhu RS, Munawer Q (2000) Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Trans. Inform. System Security* 3(2):85–106.

Phillips J (2009) How to streamline role-based access control. Accessed July 1, 2014, http://searchfinancialsecurity.techtarget.com/tip/How-to-streamline-role-based-access-control.

Raghavan P, Thompson CD (1987) Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica* 7(4):365–374.

Saltzer JH (1974) Protection and the control of information sharing in multics. *Comm. ACM* 17(7):388–402.

Sandhu R, Munawer Q (1998) How to do discretionary access control using roles. *Proc. Third ACM Workshop Role-Based Access Control* (ACM, New York), 47–54.

Sandhu R, Ferraiolo D, Kuhn R (2000) The NIST model for role-based access control: Towards a unified standard. *Proc. Fifth ACM Workshop Role-Based Access Control* (ACM, New York), 47–63.

Sandhu RS, Coyne EJ, Feinstein HL, Youman CE (1996) Role-based access control models. *Computer* 29(2):38–47.

Saviynt (2012) Information provided via letter from the CEO of Saviynt to the authors, June 4.

Schneider FB (2000) Enforceable security policies. *ACM Trans. Inform. System Security* 3(1):30–50.

Shafiq B, Masood A, Joshi J, Ghafoor A (2005) A role-based access control policy verification framework for real-time systems. *Proc. 10th IEEE Internat. Workshop Object-Oriented Real-Time Dependable Systems* (IEEE Computer Society, Washington, DC), 13–20.

Takabi H, Joshi JBD (2010) StateMiner: An efficient similarity-based approach for optimal mining of role hierarchy. *Proc. 15th ACM Sympos. Access Control Models Tech.* (ACM, New York), 55–64.

Takabi H, Joshi JBD, Ahn G (2010) Security and privacy challenges in cloud computing environments. *Security Privacy, IEEE* 8(6): 24–31.

Vaidya J, Atluri V, Warner J (2006) RoleMiner: Mining roles using subset enumeration. *Proc. 13th ACM Conf. Comput. Comm. Security* (ACM, New York), 144–153.

Vaidya J, Atluri V, Guo Q (2007) The role mining problem: Finding a minimal descriptive set of roles. *Proc. 12th ACM Sympos. Access Control Models Tech.* (ACM, New York), 175–184.

Vaidya J, Atluri V, Guo Q (2010) The role mining problem: A formal perspective. *ACM Tran. Inform. System Security* 13(3):Article 27.

Vaidya J, Atluri V, Guo Q, Adam N (2008) Migrating to optimal RBAC with minimal perturbation. *Proc. 13th ACM Sympos. Access Control Models Tech.* (ACM, New York), 11–20.

Vazirani VV (2001) *Approximation Algorithms* (Springer-Verlag, Berlin).

Xu Z, Stoller SD (2012) Algorithms for mining meaningful roles. *Proc. 17th ACM Sympos. Access Control Models Tech.* (ACM, New York), 57–66.

Yang J, Leung JY-T (2005) A generalization of the weighted set covering problem. *Naval Res. Logist.* 52(2):142–149.